

Guessing the Washington Michelin stars

Bertrand Dolimier

September 28, 2016

Executive Summary

Objective: Predict the result of the newly created Michelin guide for the Washington, DC area. The outcome will include a) the number of restaurants starred, the name of restaurants with their corresponding stars (1, 2 or 3)

Approach: Collect data for Restaurants in the other US cities where a Michelin guide already exist, explore for each of them the publicly available information, select the most relevant dataset and come up with a predictive algorithm (FYI no personal opinion just data science).

Tech stack Used R, R studio and Knit Markdown (Rmd extension) and base R libraries (httr, jsonlite & XML) to integrate documentation with code and generate the pdf doc.

The multi-step process is described below:

Base Geo. Dataset

1. Enter General area demographic in csv file with the following data point: Number of Restaurants in city, Number of Restaurants in the metro area, City Population, Metro population, Average Income and number of starred restaurants.
2. Run simple model to predict number of stars to be awarded in the Washington DC area.

Enriching the 2 datasets

1. Collect the names of currently awarded restaurants along with the number of stars for New-York, Chicago and San-Francisco.
2. To better predict a sample of non-awarded restaurants are randomly selected. They were labeled as 0 starred restaurants. All restaurants of existing Michelin US areas are saved in a .csv file and will be mentioned as “training” data.
3. Collect restaurants widely selected that are potential candidates for a Michelin award in the Washington DC area
4. Collect in the preeminent local press the professional gastronomy reviews. We will collect the number of review along with the year of the review and the number of time the restaurants was mention in the dailies archives. The news organism selected were the New-York Times, the Chicago Tribunes and the San-Francisco Chronicle.
5. Collect customers reviews and rating from the Yelp web site.

Polishing the DC dataset

1. Removing the bib gourmand from the test dataset.

Analysing the Data

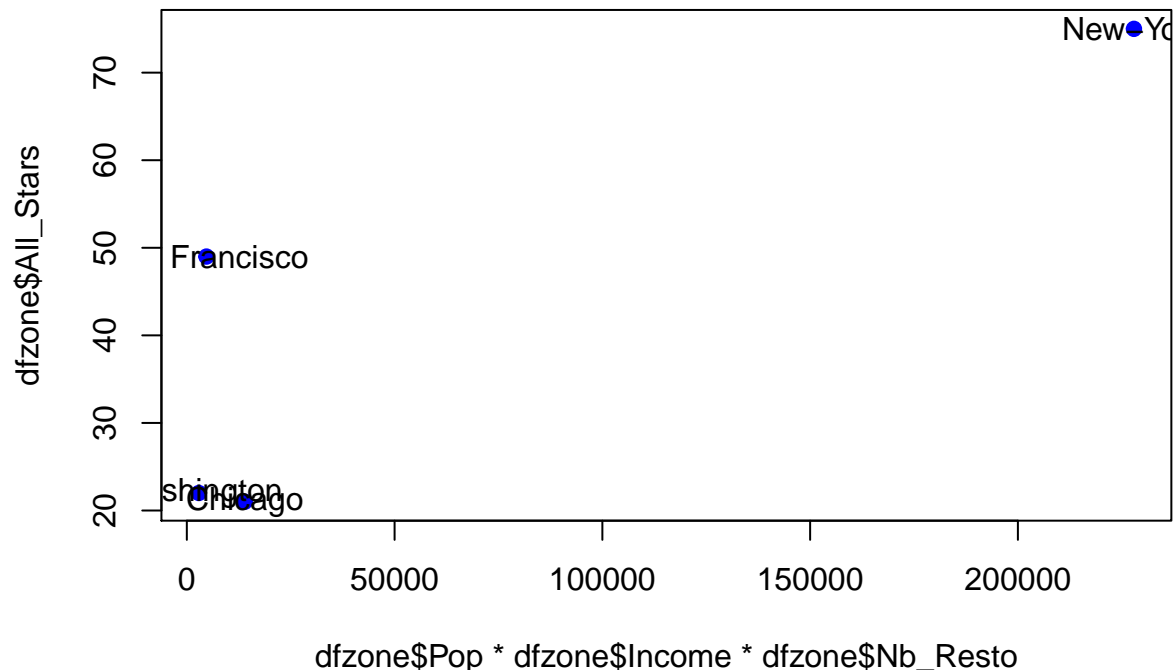
1. Use linear regression to determine a model based on the data collected above.

2. Collect restaurants from the Washington area.
3. Run the model selected on step 5 and order the restaurants.
4. Using demographic information and number of restaurant in each collective area weight by the average customer rating estimate the number of 3,2 and 1 stars restaurant to be anticipated.
5. Manually “hair-cut” the ranked list (step 7) and determine the “winners”...

Detail and code

Base Geo Dataset

The simple plot below, shows how the number of Pops in the 3 existing US Michelin guide areas have a strong correlation with the number of restaurants, however Washington has more restaurants per capita, we incorporate more quantitative demographics.



Collect other local restaurants in Existing US Michelin guide areas, and save as training dataset

```
name = paste( c( "/Users/bdolimier/persodev/michelinDC/output/usa2016_result.csv" ) )
usadata <- read.csv( name , header=TRUE, stringsAsFactors=FALSE , sep = ",")
dfusa <- as.data.frame( usadata )
dfusa <- dfusa[ -c(1) ]
myIndex <- length( dfusa[[1]])
```

```

# Take zz pages sample...
for ( zz in 1:4 ) {
  if (zz==1) zone <- "New-York"
  if (zz==2) zone <- "Chicago"
  if (zz==3) zone <- "San Francisco"
  if (zz==4) zone <- "Washington"
  zoneEncode <- URLencode( toString( zone ) )

  # Get sorted by rating the 2 price points 4 and 3
  for ( price in 3:4 ) {
    for ( page in 0:15 ) {
      start <- page * 10
      YcacheName <- paste0( "/Users/bdolimier/persodev/michelinDC/cache/" , "YELP_COLLECT_", zone, "_", page, ".csv" )

      # Reading the cache
      html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
      nbc <- nchar( as( html.raw , "character" ) )
      nbc
      # Not cached let's ask Yelp
      if ( nbc < 300 ) {
        query=paste0( "https://www.yelp.com/search?find_desc=Restaurants&find_loc=",zoneEncode,"&start=",start )
        download.file( query , destfile = YcacheName , method="curl" )
        html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
      }

      bizName <- xpathSApply(html.raw , "//span[@class='indexed-biz-name']", xmlValue)
      bizName <- gsub( "Restaurant" , "" , bizName )
      bizName <- gsub( "\n" , "" , bizName )
      bizName <- gsub( "\\." , "" , bizName )
      bizName <- gsub( " " , "" , bizName )
      nb <- length( bizName )
      if ( nb > 0 ) {
        for ( jj in 1:nb ) {
          bizName[jj] <- gsub( paste0(start+jj," ") , "" , bizName[jj] )
          myIndex = myIndex + 1
          dfusa[ myIndex ,] <- c("", "", "", 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
          dfusa$Restaurant[myIndex] <- bizName[jj]
          dfusa$Zone[myIndex] <- zone
          dfusa$Ypricing[myIndex] <- price
          print ( bizName[jj] )
        }
      }
    }
  }
}
View( dfusa )
write.csv( as.data.frame(dfusa) , file = "/Users/bdolimier/persodev/michelinDC/output/michelin2016_train.csv" )

```

Collect restaurants in the targetted area Washington DC and save as target dataset

```
name = paste( c( "/Users/bdolimier/persodev/michelinDC/output/michelin2016_washington.csv" ) )
usadata <- read.csv( name , header=TRUE, stringsAsFactors=FALSE , sep = ",")
dfusa <- as.data.frame( usadata )
dfusa <- dfusa[ -c(1) ]
myIndex <- length( dfusa[[1]])

zone <- "Washington"
zoneEncode <- URLencode( toString( zone ) )

# Get sorted by rating the 2 price points 4 and 3
for ( price in 3:4 ) {
  for ( page in 0:15 ) {
    start <- page * 10
    YcacheName <- paste0( "/Users/bdolimier/persodev/michelinDC/cache/" , "YELP_COLLECT_" , zone, "_", price, ".csv" )

    # Reading the cache
    html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
    nbc <- nchar( as( html.raw , "character" ) )
    nbc
    # Not cached let's ask Yelp
    if ( nbc < 300 ) {
      query=paste0( "https://www.yelp.com/search?find_desc=Restaurants&find_loc=",zoneEncode,"&start=",start,"&end=",start+10,"&sort=rating" )
      download.file( query , destfile = YcacheName , method="curl" )
      html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
    }

    bizName <- xpathSApply(html.raw , "//span[@class='indexed-biz-name']", xmlValue)
    bizName <- gsub( "Restaurant" , "" , bizName )
    bizName <- gsub( "\n" , "" , bizName )
    bizName <- gsub( "\\." , "" , bizName )
    bizName <- gsub( " " , "" , bizName )
    nb <- length( bizName )
    if ( nb > 0 ) {
      for ( jj in 1:nb ) {
        bizName[jj] <- gsub( paste0(start+jj," ") , "" , bizName[jj] )
        myIndex = myIndex + 1
        dfusa[ myIndex ,] <- c("", "", "", 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
        dfusa$Restaurant[myIndex] <- bizName[jj]
        dfusa$Zone[myIndex] <- zone
        dfusa$Ypricing[myIndex] <- price
        print ( bizName[jj] )
      }
    }
  }
}

View( dfusa )
write.csv( as.data.frame(dfusa) , file = "/Users/bdolimier/persodev/michelinDC/output/michelin2016_washington.csv" )
```

Get Restaurant press review from NewYork Times, Chicago Tribunes & SanFrancisco Chronicle

Using google apis Populating the fields: dfusa\$art2016, art2015, art2014, artBefore (older than 2014), dateUNKNOWN

```
## Engine id:
dcId <- "faxe8adyxke" # - Washington Post = 017019937838437061749:faxe8adyxke
nyId <- "uspap9gs9vq" # - New York Times = 017019937838437061749:uspap9gs9vq
chiId <- "duuro39maos" # - Chicago Tribunes = 017019937838437061749:duuro39maos
sfId <- "navjubg8x6w" # - San Francisco Chronicle = 017019937838437061749:navjubg8x6w

name = paste( c( "/Users/bdolimier/persodev/michelinDC/output/usa2016_training.csv" ) )
usadata <- read.csv( name , header=TRUE, stringsAsFactors=FALSE, sep = "," )
dfusa <- as.data.frame( usadata )
dfusa <- dfusa[ -c(1) ]

iimin = 1
iimax = nrow( dfusa )

for(ii in iimin:iimax ) {
  resto <- toString(dfusa$Restaurant[ii])
  restoEncode <- URLencode( toString( paste("Restaurant",resto ) ) )
  zone = toString(dfusa[ii,3])
  if ( zone == "New-York" ) engineId = nyId
  if ( zone == "Chicago" ) engineId = chiId
  if ( zone == "San Francisco" ) engineId = sfId
  print( paste( restoEncode, zone ) )
  ns <- 70 # init to large than we will query for / 7 pages

  for ( page in 0:6 ) {
    cacheName5 <- paste0( "/Users/bdolimier/persodev/michelinDC/cache/" , "C5_" , zone , "_" , ii , "_" ,
    cacheName6 <- paste0( "/Users/bdolimier/persodev/michelinDC/cache/" , "C6_" , zone , "_" , ii , "_" ,
    cf <- NULL
    cf5 <- NULL
    tryCatch( cf <- fromJSON(file( cacheName6 )) , error = function(e) cacheFlg <- FALSE , warning =
    tryCatch( cf5 <- fromJSON(file( cacheName5 )) , error = function(e) cacheFlg <- FALSE , warning =

    ## If not already cached got ask google
    if ( is.null(cf) == TRUE || is.null(cf5) == TRUE ) {
      start = (page*10)+1
      if ( start < ns ) {
        print( paste("reading a new query", ii, page) )
        query=paste0("https://www.googleapis.com/customsearch/v1?key=AIzaSyAZBzP3NEFa7Fq76ElQW5PZaIKu
        res <- GET(query)

        cf5 <- do.call("rbind", content(res)[5] ) # a dataframe
        write( toJSON(cf5) , file=cacheName5) # write in cache
        cf5 <- fromJSON(toJSON(cf5))

        cf <- do.call("rbind", content(res)[6] ) # a dataframe
        write( toJSON(cf) , file=cacheName6) # write in cache
        cf <- fromJSON(toJSON(cf))
      }
    }
  }
}
```

```

    }
  }

  # If still null skip it
  if ( toString(cf) != "" && ( !is.null(nrow(cf[[1]])) ) ) {
    nc <- nrow(cf[[1]])
    if ( is.null(nc) ) nc = 0
    ns <- as.numeric( cf5[[3]] )
    if ( length(ns) == 0 ) ns = 0
    dfusa$artTotal[ii] = ns
    if ( ns>10000 ) ns = 10000
    dfusa$artTotalCap[ii] = ns

    if ( nc > 0 ) {
      for ( jj in 1:nc ) {
        titre <- cf[[1]]$title[[jj]]
        # Date method 1
        datum <- cf[[1]]$pagemap$metatags[[jj]]$ptime
        # Date method 2
        if ( is.null(datum) ) datum <- cf[[1]]$pagemap$metatags[[jj]]$date
        # Date method 3
        if ( is.null(datum) ) datum <- cf[[1]]$pagemap$metatags[[jj]]$sailthru.date
        # Date giving up
        if ( is.null(datum) ) {
          print ( paste("no date for ", ii, page, jj))
          datum = -1
        }

        dd <- substr( toString( gsub("-", "", datum) ), 0, 4)
        print ( paste( ii , page , jj ,datum) )
        if ( dd == 2016 ) dfusa[ii,5] <- dfusa$art2016[ii]+1
        if ( dd == 2015 ) dfusa[ii,6] <- dfusa$art2015[ii]+1
        if ( dd == 2014 ) dfusa[ii,7] <- dfusa$art2014[ii]+1
        if ( dd < 2014 ) dfusa[ii,8] <- dfusa$artBefore[ii]+1 # older than 2014
        if ( dd == -1 ) dfusa[ii,9] <- dfusa$dateUNKNOWN[ii]+1 # date unknown
      }
    }
  }
}

View( dfusa )
write.csv( as.data.frame(dfusa) , file = "/Users/bdolimier/persodev/michelinDC/output/michelin2016_train" )

```

Get Restaurant press review from the Washington Post.

Populating the fields: dfusa\$art2016, art2015, art2014, artBefore (older than 2014), dateUNKNOWN

Yelp review and Classification

Populating Yreview, Yrating, Ypricing

Get google rating and neighborhood

Populating: Zreview, Zrating, Neighborhood

Zagat review through google apis

Populating fields: Fscore, Dscore, Sscore, type, buzz, buzz2, Neighborhood

```
#name = paste( c( "/Users/bdolimier/persodev/michelinDC/output/usa2016_result.csv" ) )
name = paste( c( "/Users/bdolimier/persodev/michelinDC/output/michelin2016_washington.csv" ) )
usadata <- read.csv( name , header=TRUE, stringsAsFactors=FALSE , sep = "," )
dfusa <- as.data.frame( usadata )
dfusa <- dfusa[ -c(1) ]

iimin <- 1
iimax = nrow( dfusa )

for(ii in iimin:iimax ) {
  print(ii)
  dfusa$Fscore[ii] <- 0
  dfusa$Dscore[ii] <- 0
  dfusa$Sscore[ii] <- 0
  dfusa$type[ii] <- " "
  dfusa$buzz[ii] <- " "
  dfusa$buzz2[ii] <- " "
  dfusa$Neighborhood[ii] <- " "

  resto <- toString(dfusa[ii,1])
  zone = toString(dfusa[ii,3])
  zone <- URLencode( toString( zone ) )

  # Reading the cache
  YcacheName <- paste0( "/Users/bdolimier/persodev/michelinDC/cache/" , "ZAGAT_" , zone , "_" , ii , ".html" )
  html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
  nb <- nchar( as( html.raw , "character" ) )
  print(nb)
  # Not cached let's ask Zagat
  if ( nb < 300 ) {
    restoEncode <- URLencode( toString( resto ) )
    restoEncode <- URLencode( paste( "Restaurant+" , toString( resto ) , "+Washington" ) )
    query <- paste0("https://www.zagat.com/search/place?text=",restoEncode,"&session=62981284&session=")
    download.file( query , destfile = YcacheName , method="curl" )
    html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
    redirect <- xpathSApply( html.raw , "//a" , xmlGetAttr, 'href' )

    if ( length( redirect ) < 2 ) {
      download.file( redirect , destfile = YcacheName , method="curl" )
      html.raw<-htmlTreeParse( YcacheName, useInternalNodes=T )
    }
  }

  bizName <- xpathSApply(html.raw , "//script" , xmlValue)
```

```

for ( vv in 1:length( bizName ) ) {
  if ( length(grep( "zga.setCustomVar" , bizName[vv])) > 0 ) {
    customV <- unlist(strsplit(bizName[vv], split="\n"))
    customV <- gsub( "zga.setCustomVar" , "" , customV )
    customV <- gsub( " " , "" , customV )
    customV <- gsub( ";" , "" , customV )
    customV <- gsub( ")" , "" , customV )
    customV <- gsub( '"' , "" , customV )
    for ( zz in 1:length( customV ) ) {
      typeV <- unlist(strsplit( customV[zz], split=","))
      if ( length(grep( "FoodScore" , typeV[1] ) ) ) dfusa$Fscore[ii] <- as.numeric(unlist(strsplit(customV[zz], split=","))[1])
      if ( length(grep( "DecorScore" , typeV[1] ) ) ) dfusa$Dscore[ii] <- as.numeric(unlist(strsplit(customV[zz], split=","))[1])
      if ( length(grep( "ServiceScore" , typeV[1] ) ) ) dfusa$Sscore[ii] <- as.numeric(unlist(strsplit(customV[zz], split=","))[1])
      if ( length(grep( "placeType" , typeV[1] ) ) ) dfusa$type[ii] <- typeV[2]
      if ( length(grep( "appearsInBuzz" , typeV[1] ) ) ) dfusa$buzz[ii] <- typeV[2]
      if ( length(grep( "appearsInLists" , typeV[1] ) ) ) dfusa$buzz2[ii] <- typeV[2]
      if ( length(grep( "placeNeighborhood" , typeV[1] ) ) ) dfusa$Neighborhood[ii] <- typeV[2]
      print( typeV[2] )
    }
  }
}
}
View( dfusa )
write.csv( as.data.frame(dfusa) , file = "/Users/bdolimier/persodev/michelinDC/output/michelin2016_wash" )

```

Remove bib gourmand

Removed the already awarded “Bib gourmand” restaurants.

Analyse the data

Fields collected are:

1. Restaurant: Name of the Restaurant
2. Chef: Name of the Chef (not always present)
3. Zone: New-York, Chicago, SF and DC
4. Etoile: 0 to 3 Stars
5. art2016: Number of mention in select press for the zone in 2016
6. art2015: Number of mention in select press for the zone in 2015
7. art2014: Number of mention in select press for the zone in 2014
8. artBefore: Number of mention in select press for the zone before 2014
9. dateUNKNOWN: Number of mention in select press for the zone date unknown
10. artTotal: Number of mention in select press for the zone (all dates)
11. Yreview: Number of Yelp reviews
12. Yrating: Aggregated Yelp review
13. Ypricing: Yelp pricing level (“\$” to “\$\$\$\$”)
14. Zrating: Zagat rating
15. Zpricing: Zagat pricing level
16. Zbuzz: Zagat buzz indicator

17. Zbuzz: Zagat buzz indicator 2
18. Neighborhood: Restaurant Neighborhood

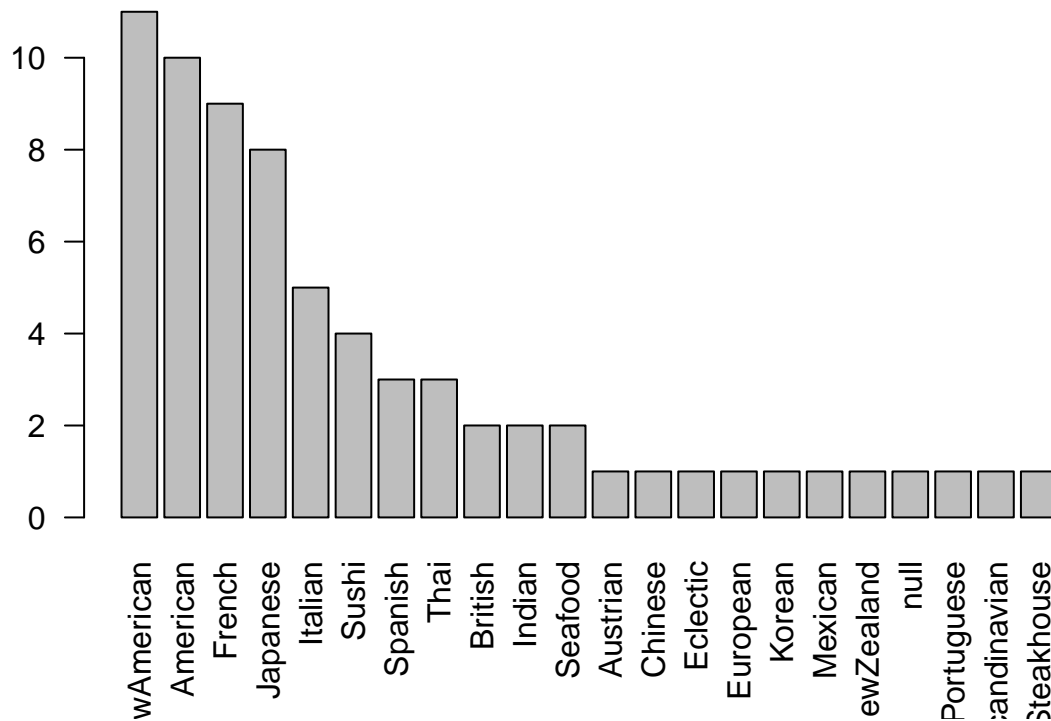
Create Mode / Plot Linear Regression

```
# Training data collected
training = paste( c( "/Users/bdolimier/persodev/michelinDC/output/michelin2016_training2.csv" ) )
usadata <- read.csv( training , header=TRUE, stringsAsFactors=FALSE , sep = ",")
dfusa <- as.data.frame( usadata )
dfusa <- dfusa[ -c(1) ]
dfusa["NumericType"] <- 0

# Test data to apply prediction on
wash = paste( c( "/Users/bdolimier/persodev/michelinDC/output/michelin2016_washington.csv" ) )
dcdata <- read.csv( wash , header=TRUE, stringsAsFactors=FALSE , sep = ",")
dfdc <- as.data.frame( dcdata )
dfdc <- dfdc[ -c(1) ]
dfdc["NumericType"] <- 0

# Randk passed stored restaurant type
fit <- lm( Etoile ~ factor(type) , na.action=na.exclude ,data = dfusa )
ndf <- subset( dfusa, dfusa$Etoile > 0 )
ndf <- subset( ndf, (ndf$type == "" ) == FALSE , na.action=na.exclude, null.action=null.exclude )
ndf <- subset( ndf, table(ndf$type) > 0 )

ndf <- table(ndf$type)
ndf <- ndf[order(-ndf)]
barplot(ndf,las=2)
```



```

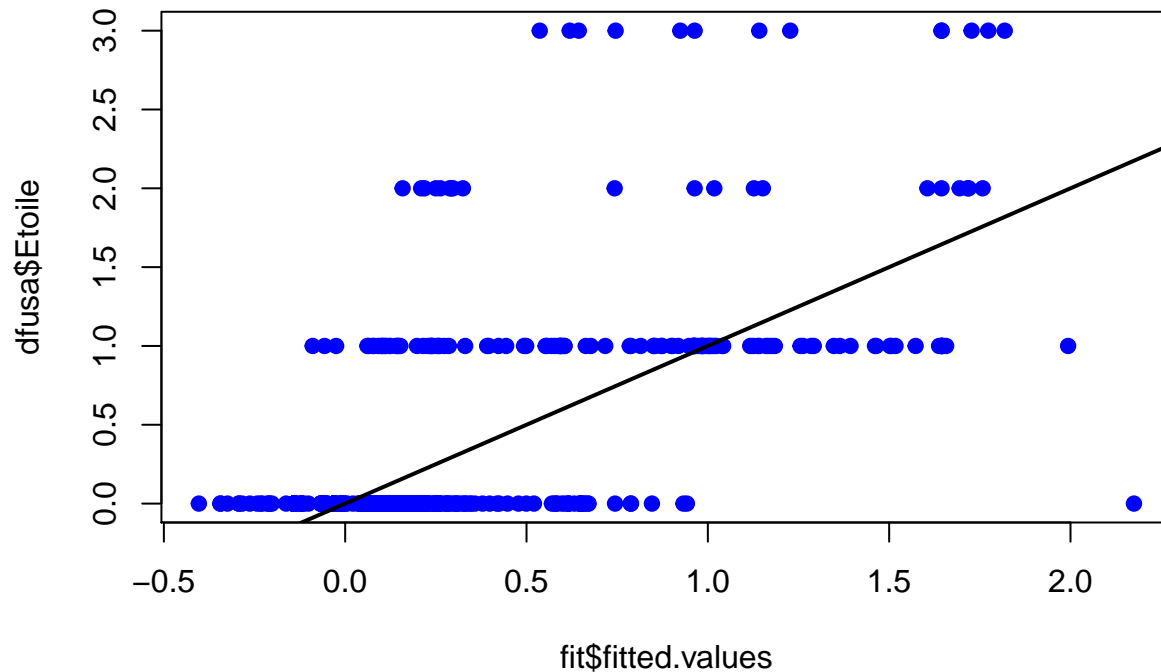
# Apply numerical value to restaurant type
iimin=1
iimax = nrow( dfdc )
for(ii in iimin:iimax ) {
  done <- FALSE
  for ( vv in 1:10 ) {
    if( done == FALSE & length(grep( "LatinAmerican" , dfdc$type[ii]))==1 ) { dfdc$NumericType[ii] <- 0
    if( done == FALSE & length(grep( "American(New)" , dfdc$type[ii]))==1 ) { dfdc$NumericType[ii] <- 0
    if( done == FALSE & length(grep( names(ndf[vv]) , dfdc$type[ii]))==1 ) { dfdc$NumericType[ii] <- 0
  }
  if( done == FALSE ) dfdc$NumericType[ii] <- 0
}
iimin=1
iimax = nrow( dfusa )
for(ii in iimin:iimax ) {
  done <- FALSE
  for ( vv in 1:10 ) {
    if( done == FALSE & length(grep( "LatinAmerican" , dfusa$type[ii]))==1 ) { dfusa$NumericType[ii] <- 0
    if( done == FALSE & length(grep( "American(New)" , dfusa$type[ii]))==1 ) { dfusa$NumericType[ii] <- 0
    if( done == FALSE & length(grep( names(ndf[vv]) , dfusa$type[ii]))==1 ) { dfusa$NumericType[ii] <- 0
  }
  if( done == FALSE ) dfusa$NumericType[ii] <- 0
}

par(mfrow=c(1,1))

fit <- lm( Etoile ~ artTotalCap *Yreview * Yrating * Ypricing * NumericType , na.action=na.exclude ,data=dfusa)

plot( fit$fitted.values , dfusa$Etoile ,pch=19,col="blue")
abline(lm( dfusa$Etoile ~ fit$fitted.values ), lwd = 2)

```



```

ndf <- subset( dfdc, dfdc$Yreview > 5 )
mesEtoiles <- predict( fit , ndf, interval="predict" )
mesEtoiles <- cbind( mesEtoiles, ndf$Restaurant )

# Ordering the results
mesEtoiles <- mesEtoiles[order(mesEtoiles[,1], decreasing = TRUE ) ,]

stared <- as.data.frame( read.csv( "/Users/bdolimier/persodev/michelinDC/output/michelin2016_template.csv" ) )

# Extracting and
stared$restaurant <- as.data.frame(mesEtoiles[ 1:23 , 4])[,1]
stared[,2] <- c(3,3,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 )
write.csv( stared , file = "/Users/bdolimier/persodev/michelinDC/BertrandDolimier.csv", row.names=TRUE )

```

Results are stored in file BertrandDolimier.csv