Fitcoach package workflow example

Niraj Juneja, Charles de Lassence

March 12, 2016

Example 1 : Data Loader- Getting data from Fitbit API

This part explains how to connect to the Fitbit API and get your data, using DataLoader.

Step 1: You first need to make sure that you have registered an app and set it as *Personal* in order to retrieve intraday data. You will need the following credentials in order to connect the API: App name (or *OAuth 2.0 Client ID*), Client Key and Client Secret.

Step 2: We initialize a new DataLoader object, and connect to the API with Oauth2, using the credentials described above.

```
mydata <- DataLoader$new()
mydata$connect(appname = "cdlr",
               key = "227FWR",
               secret = "3089e3d1ac5dde1aa00b54a0c8661f42"
)
```

Step 3: We request the data and write it to JSON files using the `request` method. You need to specify the type of timeseries ('day' or 'intraday'), the list of activities (full list here), the start and end dates, and the folder in which the JSON files will be written.

```
masterPath <- system.file("extdata",
                          "daily-time-series",
                          package = "fitcoach")

mydata$request(
    type = 'day',
    activities = list("calories", "steps", "distance", "minutesVeryActive"),
    start.date = "2016-01-01",
    end.date = "2016-02-01",
    path = masterPath
)
```

Once the JSON files have been created, they can be used for further analysis.

Example 2 : Fit Analyzer - Daily File Analysis

Examples below demonstrate usage scenarios for FitAnalyzer

Step 1: We first need to point to a folder that contains the Json files for "daily" file analysis. Refer These files are created by DataLoader.R

We then create a new instance of Fitanalyzer passing in the folder and the goal that we want to optimize on. Goals can be the following a) calories b)steps c) distance d)floors

The example below uses *steps* as the goal

```r
masterPath <- system.file("extdata",
                          "daily-time-series",
                          package = "fitcoach")

ana <- FitAnalyzer$new("steps")
```

Step 2: Next we get the data.frame ready for analysis. Note this data.frame is cleaned and augmented with additional data elements not present in the json file. eg: we augment weekday, weekend and mark rows that are valid

```r
timeseries.frame <-
    ana$getAnalysisFrame(folder = masterPath,
                         analysis.type = "daily")
head(timeseries.frame)
```

```
##           date calories caloriesBMR steps distance floors elevation
## 898 2015-07-31     1867        1759   195  0.14062      0         0
## 899 2015-08-01     3245        1758 12866 10.44119     13        39
## 900 2015-08-02     2867        1758  5023  3.65184     11        33
## 901 2015-08-03     2982        1758 10112  7.35157      6        18
## 902 2015-08-04     2734        1758  5725  4.16213      7        21
## 903 2015-08-05     3012        1758  9155  6.72913      5        15
##     minutesSedentary minutesLightlyActive minutesFairlyActive
## 898              205                   10                   0
## 899              672                  269                   5
## 900              691                  168                  25
## 901             1161                  143                   5
## 902              836                  150                  18
## 903              640                  267                  16
##     minutesVeryActive activityCalories valid   weekday weekend
## 898                 0               51  TRUE    Friday   FALSE
## 899                37             1600  TRUE  Saturday    TRUE
## 900                35             1196  TRUE    Sunday    TRUE
## 901                66             1253  TRUE    Monday   FALSE
## 902                23              961  TRUE   Tuesday   FALSE
## 903                16             1372  TRUE Wednesday   FALSE
```

Step 3: next we find the most important variables that are enabling meeting the goals for the person. Note this call creates a glm model behind the scenes and ranks the variables based on the coefficients of the glm model. You can also get the glm fit object to do further analysis

```r
vars <- ana$findImportantVariables(tsDataFrame = timeseries.frame,
                                   seed = 12345)
vars
```

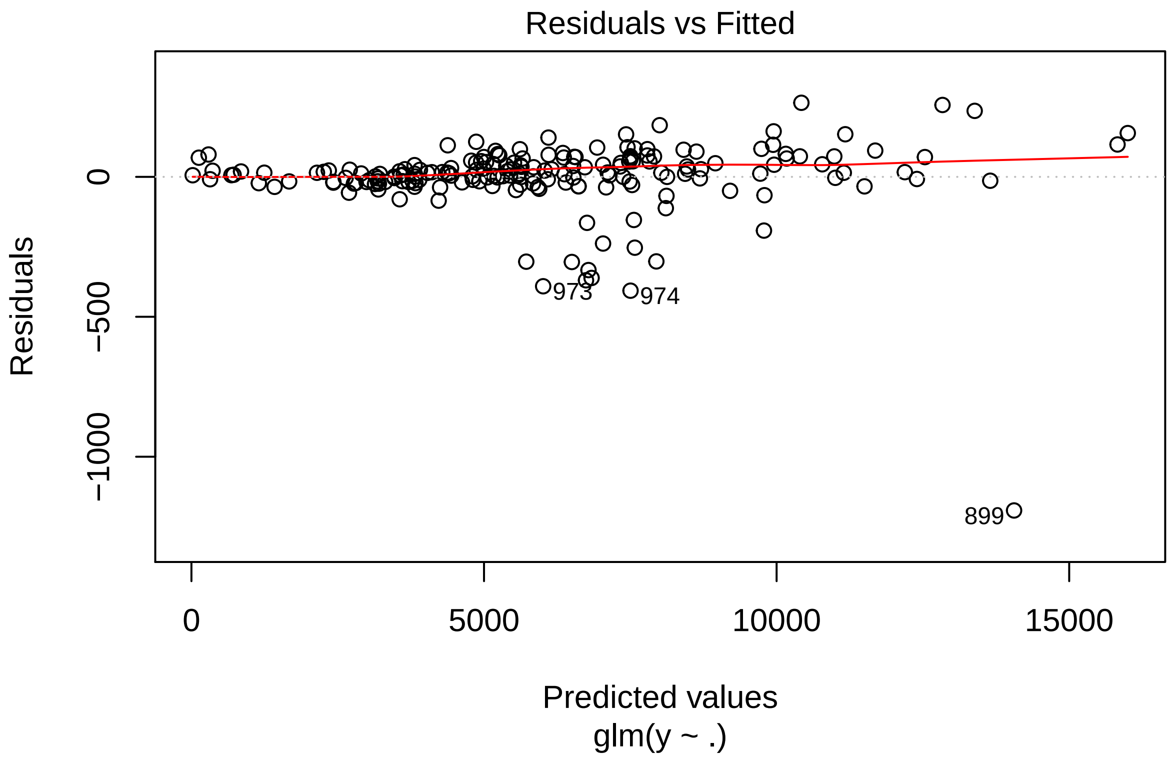
```
##      Overall                 name
## 1 83.7286565             distance
## 2  3.4383872 minutesLightlyActive
## 3  1.8757480               floors
## 4  1.8523923            elevation
## 5  1.2891208  minutesFairlyActive
## 6  1.1386025     minutesSedentary
## 7  1.1244243    minutesVeryActive
## 8  0.2122723              holiday
```

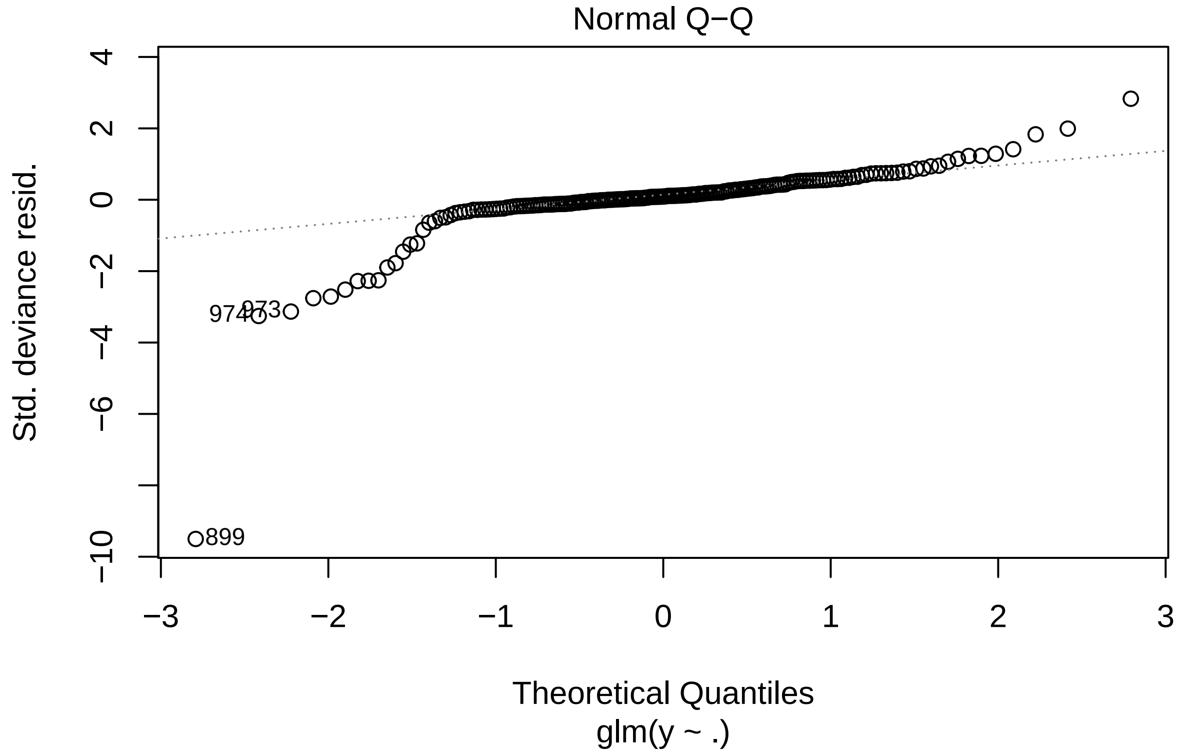
Getting the fit object

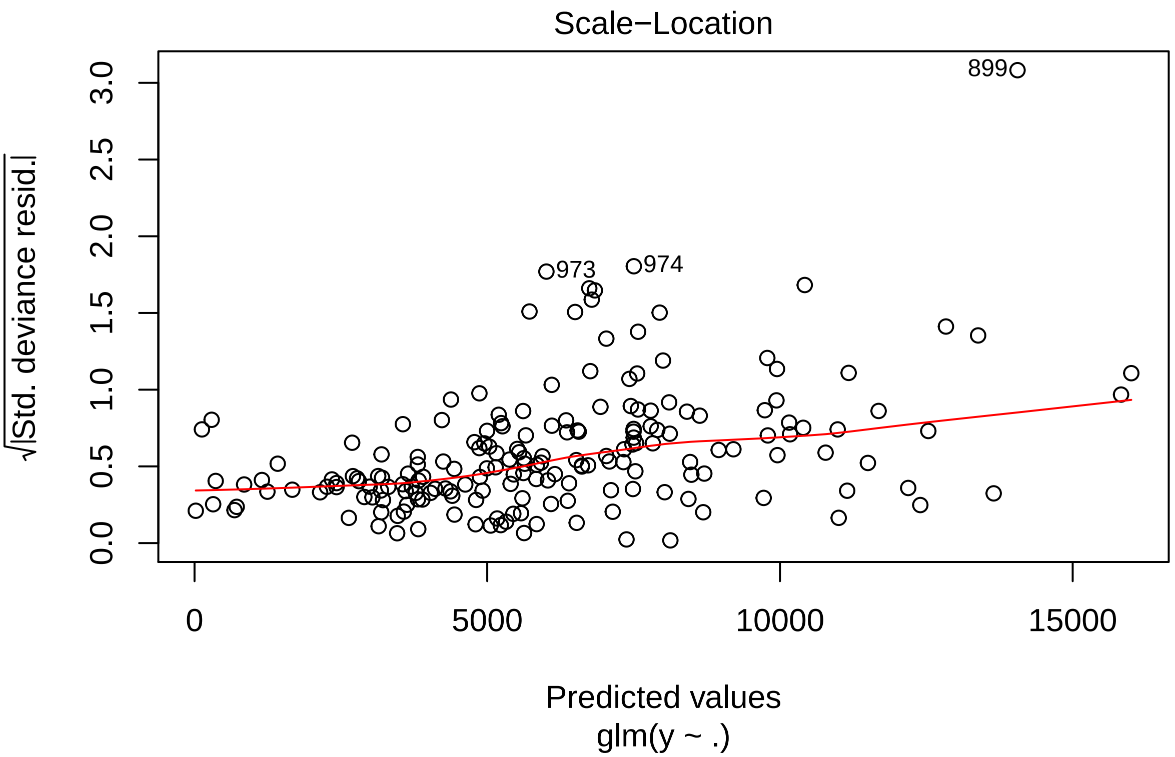
```
fit <- ana$getFit()
summary(fit)
```

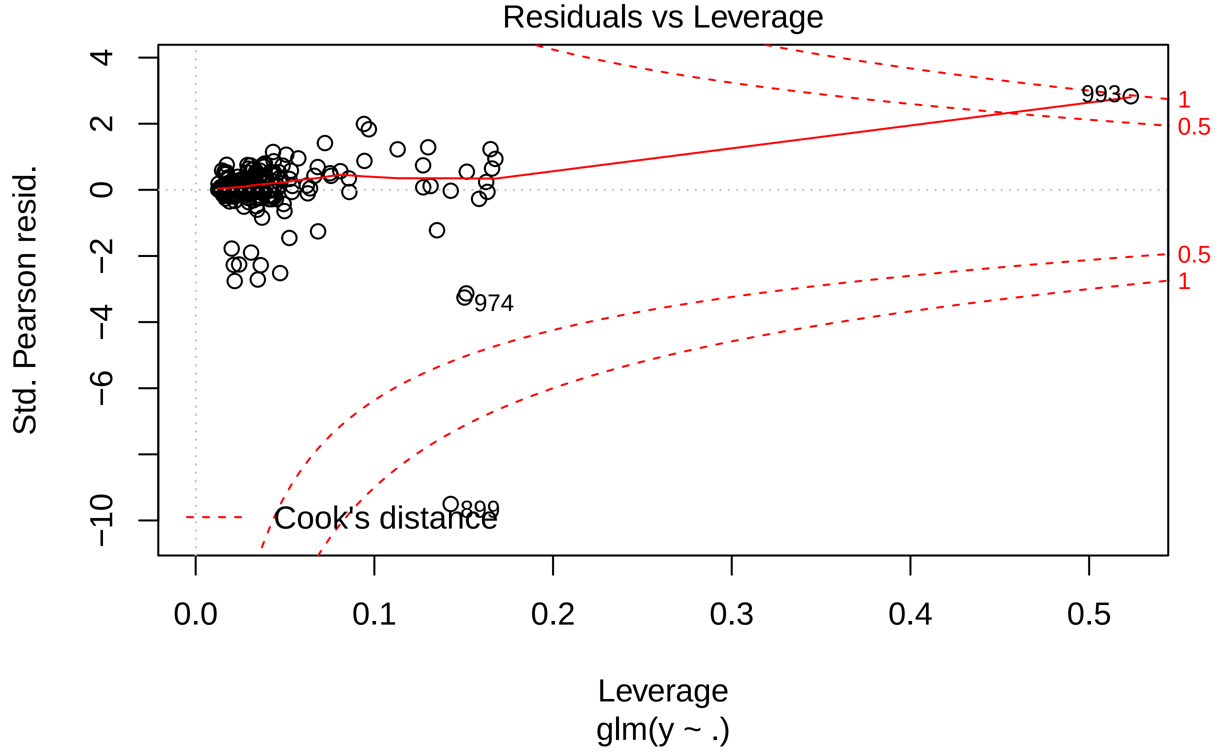
```
##
## Call:
## glm(formula = y ~ ., family = "gaussian", data = x)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1192.03    -17.96     12.12     54.98    264.72
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -83.72528   62.25542  -1.345 0.180342
## distance            1323.59328   15.80813  83.729  < 2e-16 ***
## floors              -472.34421  251.81645  -1.876 0.062291 .
## elevation            153.87808   83.06992   1.852 0.065589 .
## minutesSedentary       0.05395    0.04738   1.139 0.256365
## minutesLightlyActive   1.30032    0.37818   3.438 0.000725 ***
## minutesFairlyActive    1.70467    1.32235   1.289 0.198992
## minutesVeryActive      1.66153    1.47767   1.124 0.262314
## holiday                5.05059   23.79298   0.212 0.832132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 18358.87)
##
##     Null deviance: 1752078977  on 190  degrees of freedom
## Residual deviance:    3341314  on 182  degrees of freedom
## AIC: 2428
##
## Number of Fisher Scoring iterations: 2
```

```
plot(fit)
```


Residuals vs Fitted
Residuals
0
−500
−1000
973
974
899
0
5000
10000
15000
Predicted values
glm(y ~ .)

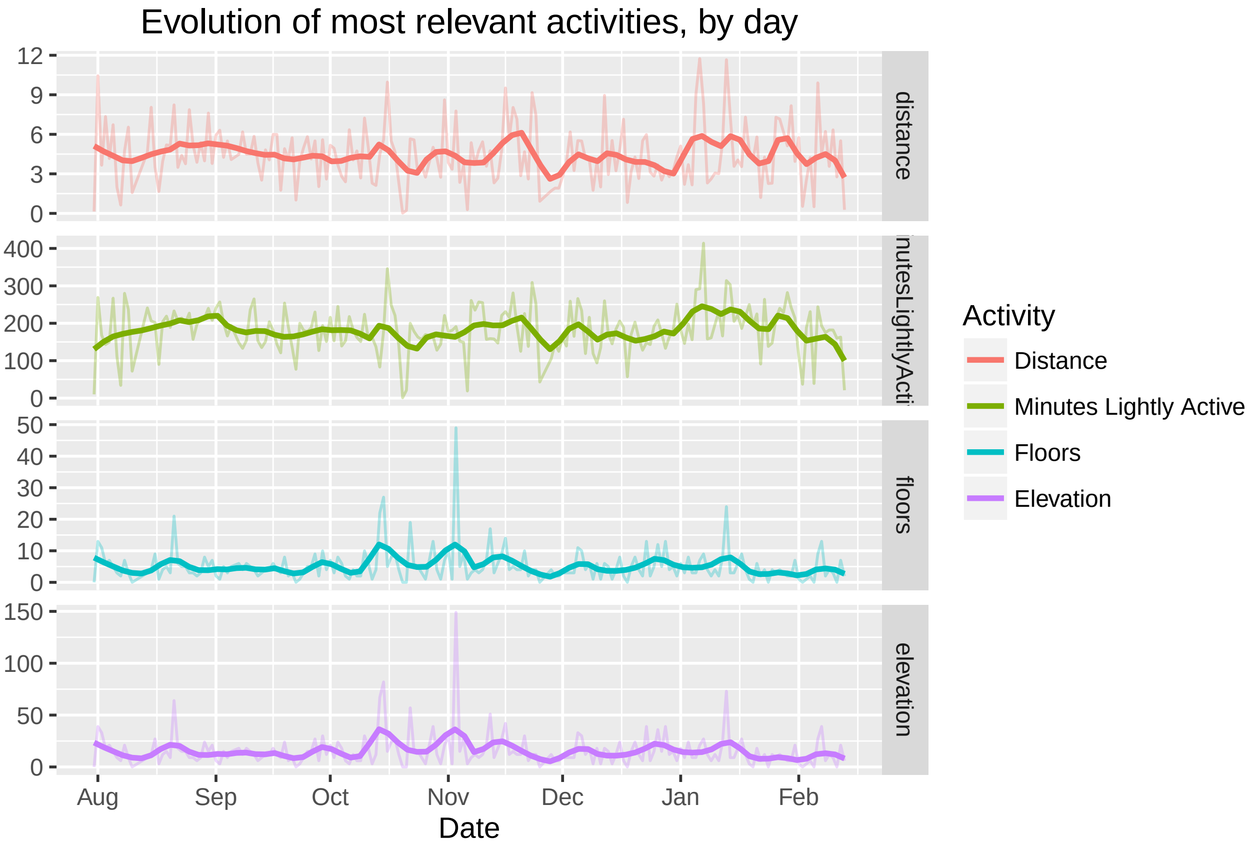

Normal Q−Q
Std. deviance resid.
4
2
0
−2
−4
−6
−10
974
973
899
−3
−2
−1
0
1
2
3
Theoretical Quantiles
glm(y ~ .)





Step 4: Next we can then plot the performance of the individual relative to the most important variables that are making a difference.

```
ana$showMostImportantCharts(tsDataFrame = timeseries.frame)
```



Step 5: We can also get the prediction on goal performance using the call below

```
rows.test <- timeseries.frame[sample(1:191 , 1), ]
x <- createDependentVariableFrame(master = rows.test, goal = "steps")
res <- ana$predictGoal(x)
cat(paste("Prediction for the day" , ": expected steps = ", round(res)))
```

```
## Prediction for the day : expected steps =  7481
```

Example 3 :FitAnalyzer - Intra-day File Analysis

Examples below demonstrate usage scenarios for FitAnalyzer for **Intra day analysis**

Step 1: We first need to point to a folder that contains the Json files for *intraday* file analysis. Refer These files are created by DataLoader.R

We then create a new instance of Fitanalyzer passing in the folder and the goal that we want to optimize on. Goals can be the following a) calories b)steps c) distance d)floors

The example below uses *calories* as the goal

```
masterPath <-
    system.file("extdata", "intra-daily-timeseries", package = "fitcoach")
ana <- FitAnalyzer$new("calories")
```

Step 2: Next we get the data.frame ready for analysis. Note this data.frame is cleaned and augmented with additional data elements not present in the json file. eg: we augment cumulative sum during the day,weekday, weekend etc.

```
intra <- ana$getAnalysisFrame(folder = masterPath, analysis.type = "intra.day")
head(intra)
```

```
##         date calories intra.level intra.mets intra.calorie timeseq steps
## 1 2015-12-10     2491           0        150       18.5565       1  5319
## 2 2015-12-10     2491           0        150       18.5565       2  5319
## 3 2015-12-10     2491           0        150       18.5565       3  5319
## 4 2015-12-10     2491           0        150       18.5565       4  5319
## 5 2015-12-10     2491           0        150       18.5565       5  5319
## 6 2015-12-10     2491           0        150       18.5565       6  5319
##   intra.steps floors intra.floors elevation intra.elevation distance
## 1           0      6            0        18               0  3.89512
## 2           0      6            0        18               0  3.89512
## 3           0      6            0        18               0  3.89512
## 4           0      6            0        18               0  3.89512
## 5           0      6            0        18               0  3.89512
## 6           0      6            0        18               0  3.89512
##   intra.distance weekday weekend  slot cumsum.calorie cumsum.steps
## 1              0       5       0 night        18.5565            0
## 2              0       5       0 night        37.1130            0
## 3              0       5       0 night        55.6695            0
## 4              0       5       0 night        74.2260            0
## 5              0       5       0 night        92.7825            0
## 6              0       5       0 night       111.3390            0
##   cumsum.level cumsum.mets cumsum.distance cumsum.floors cumsum.elevation
## 1            0         150               0             0                0
## 2            0         300               0             0                0
## 3            0         450               0             0                0
## 4            0         600               0             0                0
## 5            0         750               0             0                0
## 6            0         900               0             0                0
```

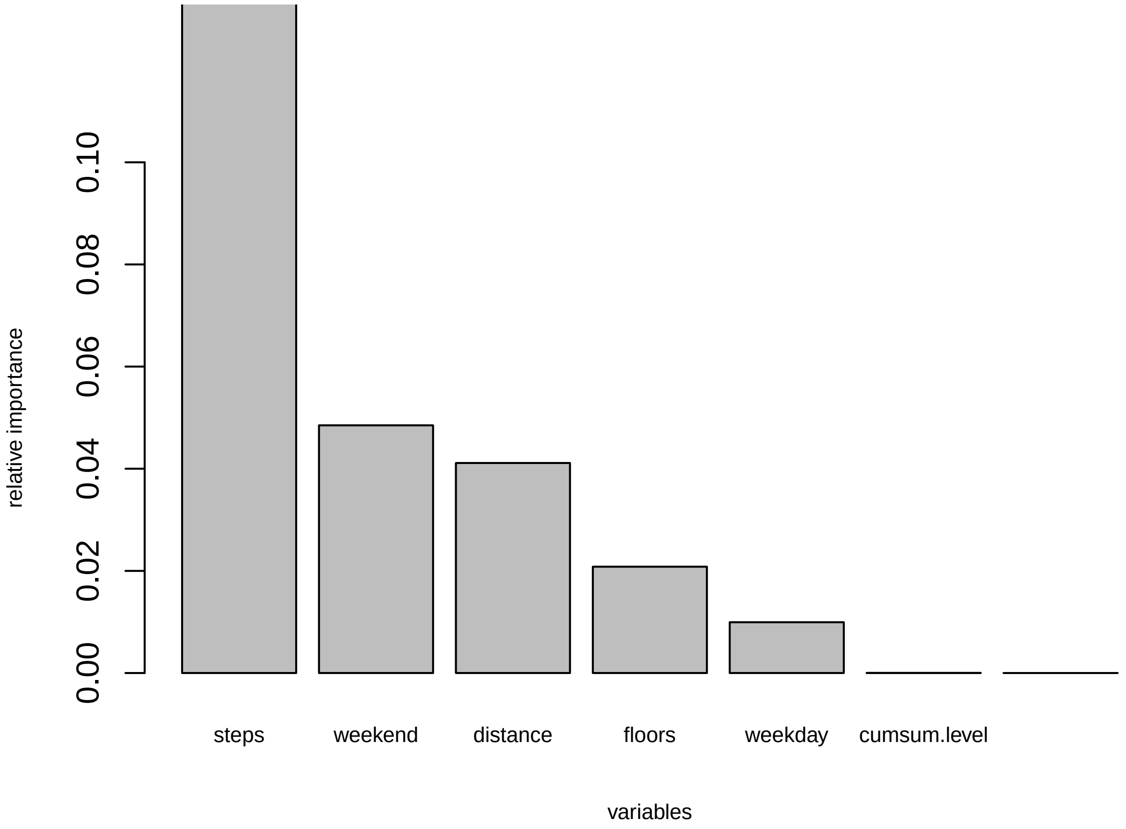
Step 3: next we find the most important variables that are enabling meeting the goals for the person. Note:this call creates a **gbm** model behind the scenes and ranks the variables based on *relative.influence* call to gbm model. You can also get the gbm fit object to do further analysis

```
vars <- ana$findImportantVariables(intra)
vars <- sort(vars, decreasing = TRUE)
vars
```

```
##        steps       weekend      distance        floors
## 1.000000e+00  4.850034e-02  4.112410e-02  2.081519e-02
##      weekday  cumsum.level cumsum.floors       timeseq
## 9.944454e-03  2.865114e-05  8.797090e-08  8.763791e-08
##     cumsum.steps  cumsum.distance   cumsum.calorie cumsum.elevation
##     4.113028e-08     1.453128e-08     1.427501e-08     7.307049e-09
##    intra.calorie      intra.steps     intra.floors       intra.mets
##     6.349336e-09     9.373329e-10     3.419370e-10     2.913440e-10
##   intra.distance             slot      intra.level        elevation
##     8.783386e-11     6.107080e-11     0.000000e+00     0.000000e+00
##  intra.elevation      cumsum.mets
##     0.000000e+00     0.000000e+00
```

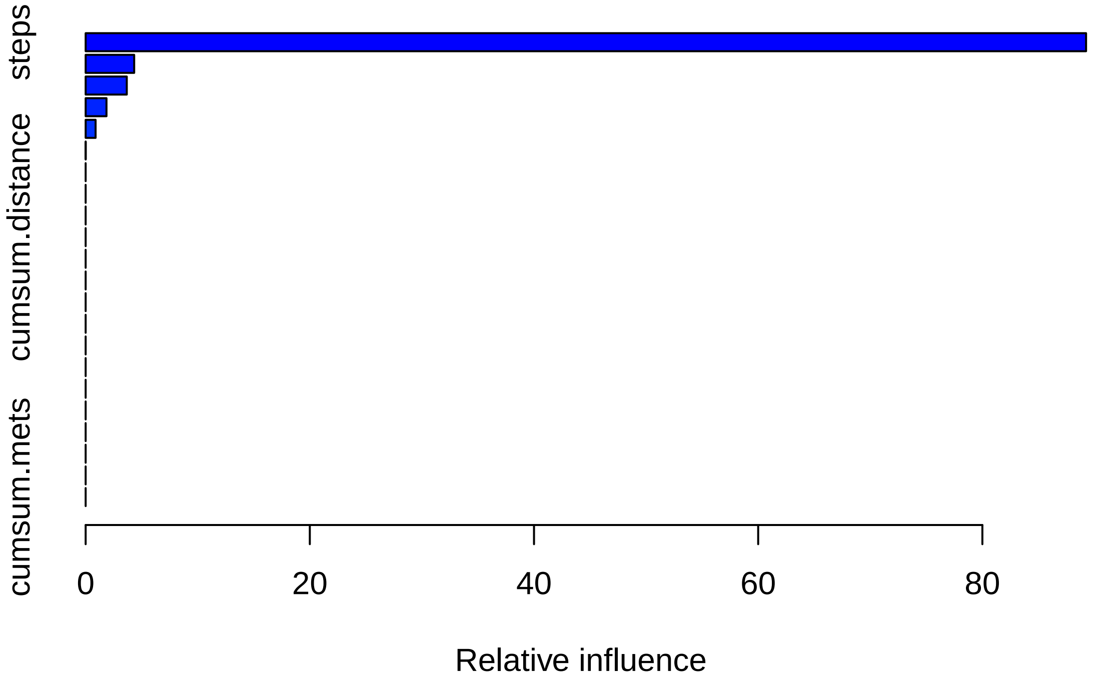
Plot of important variables below

```
vars.frame <- data.frame(variables = names(vars), values = vars)
vars.frame$lnvalue <- log(vars.frame$values)
vars.frame <- vars.frame[1:7, ]
barplot(vars.frame$value, xlab = "variables", ylab = "relative importance",
        names.arg = vars.frame$variables,
        cex.names = 0.65, cex.lab = 0.65, ylim = c(0.0, 0.1))
```



Summary of GBM model fit below

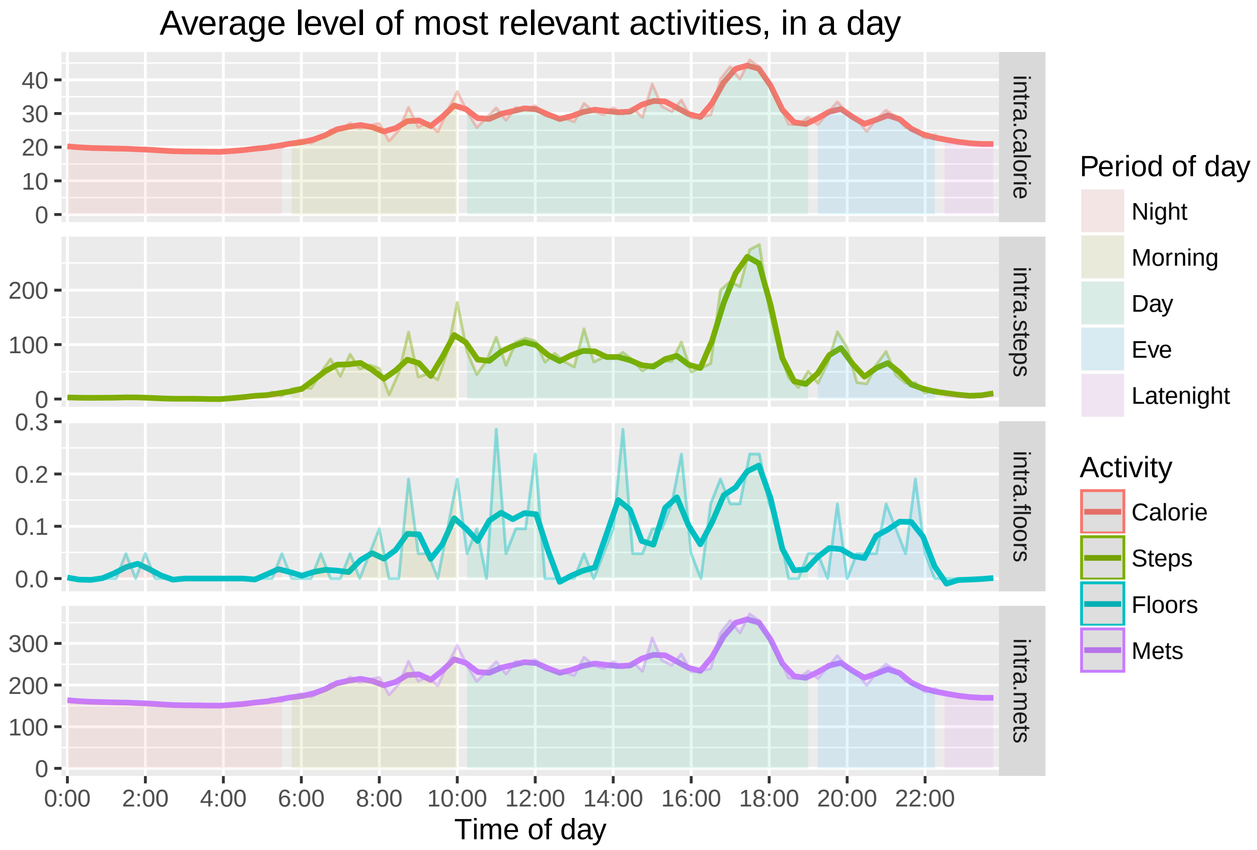
```
fit <- ana$getFit()
summary(fit)
```



```
##                                  var      rel.inf
## steps                          steps 8.925280e+01
## weekend                      weekend 4.328791e+00
## distance                    distance 3.670441e+00
## floors                        floors 1.857814e+00
## weekday                      weekday 8.875704e-01
## cumsum.level            cumsum.level 2.557195e-03
## cumsum.floors          cumsum.floors 7.851649e-06
## timeseq                      timeseq 7.821929e-06
## cumsum.steps            cumsum.steps 3.670992e-06
## cumsum.distance      cumsum.distance 1.296958e-06
## cumsum.calorie        cumsum.calorie 1.274084e-06
## cumsum.elevation cumsum.elevation 6.521746e-07
## intra.calorie          intra.calorie 5.666960e-07
## intra.steps              intra.steps 8.365959e-08
## intra.floors            intra.floors 3.051884e-08
## intra.mets                intra.mets 2.600326e-08
## intra.distance        intra.distance 7.839418e-09
## slot                            slot 5.450740e-09
## intra.level              intra.level 0.000000e+00
## elevation                  elevation 0.000000e+00
## intra.elevation      intra.elevation 0.000000e+00
## cumsum.mets              cumsum.mets 0.000000e+00
```

Step 4: Next we can then plot the performance of the individual relative to the most important variables that are making a difference. For the 4 most important variables, the average value for every 15 min of a day is plotted, along with the moving average (using `geom_smooth` from `ggplot2`),

```
ana$showMostImportantCharts(tsDataFrame = intra)
```



Step 5: We can also get the prediction on goal performance using the call below

```
rows.test <- intra[sample(1:191 , 1), ] # take any random input for test
res <- ana$predictGoal(rows.test)
cat(paste("Prediction for the day" , " : expected calories = ", round(res)))
```

```
## Prediction for the day  : expected calories =  2517
```

Example 4 :FitUtil - illustration for usage of fitutil functions

Approach to get a clean data.frame from json files

```
# masterPath is the folder containing Json files
masterPath <- system.file("extdata", "daily-time-series", package = "fitcoach")

# Create the data.frame. This is not cleaned
master <- createTsMasterFrame(masterPath)

# Identify and Mark rows that are valid. i.e distance for the day >0
master <- markValidRows(master)

# Filter Valid rows only
master <- master[master$valid == TRUE, ]

# Augment data with additional information. Eg: weekday information
```

```
master <- augmentData(master)
head(master)
```

```
##           date calories caloriesBMR steps distance floors elevation
## 898 2015-07-31     1867        1759   195  0.14062      0         0
## 899 2015-08-01     3245        1758 12866 10.44119     13        39
## 900 2015-08-02     2867        1758  5023  3.65184     11        33
## 901 2015-08-03     2982        1758 10112  7.35157      6        18
## 902 2015-08-04     2734        1758  5725  4.16213      7        21
## 903 2015-08-05     3012        1758  9155  6.72913      5        15
##     minutesSedentary minutesLightlyActive minutesFairlyActive
## 898              205                   10                   0
## 899              672                  269                   5
## 900              691                  168                  25
## 901             1161                  143                   5
## 902              836                  150                  18
## 903              640                  267                  16
##     minutesVeryActive activityCalories valid   weekday weekend
## 898                 0               51  TRUE    Friday   FALSE
## 899                37             1600  TRUE  Saturday    TRUE
## 900                35             1196  TRUE    Sunday    TRUE
## 901                66             1253  TRUE    Monday   FALSE
## 902                23              961  TRUE   Tuesday   FALSE
## 903                16             1372  TRUE Wednesday   FALSE
```