R Recap Session - IE Big Data Club

Felix Mueller

2/15/2017

R Sessions

- How to apply models in general
- Troubleshooting
- Preparation
- simple data transformations
- Data wrangling with dplyr
- Functions
- Split your data
- Apply simple model
- Q & A
- Plan for the future

How to apply models in general

- Understanding the problem
- Understanding the data
- Preparing the data
- Split in Train & Test
- Model
- Tuning of the model
- Evaluation

Troubleshooting

- ask R: ?function_i_want_to_apply
- ask R: args(function_i_want_to_apply)
- Google: “R Error message”
- Stackoverflow
- Cheat Sheets
- Kaggle
- Starting point: first error in script
- Error might occur much later than the reason for it

Before you start

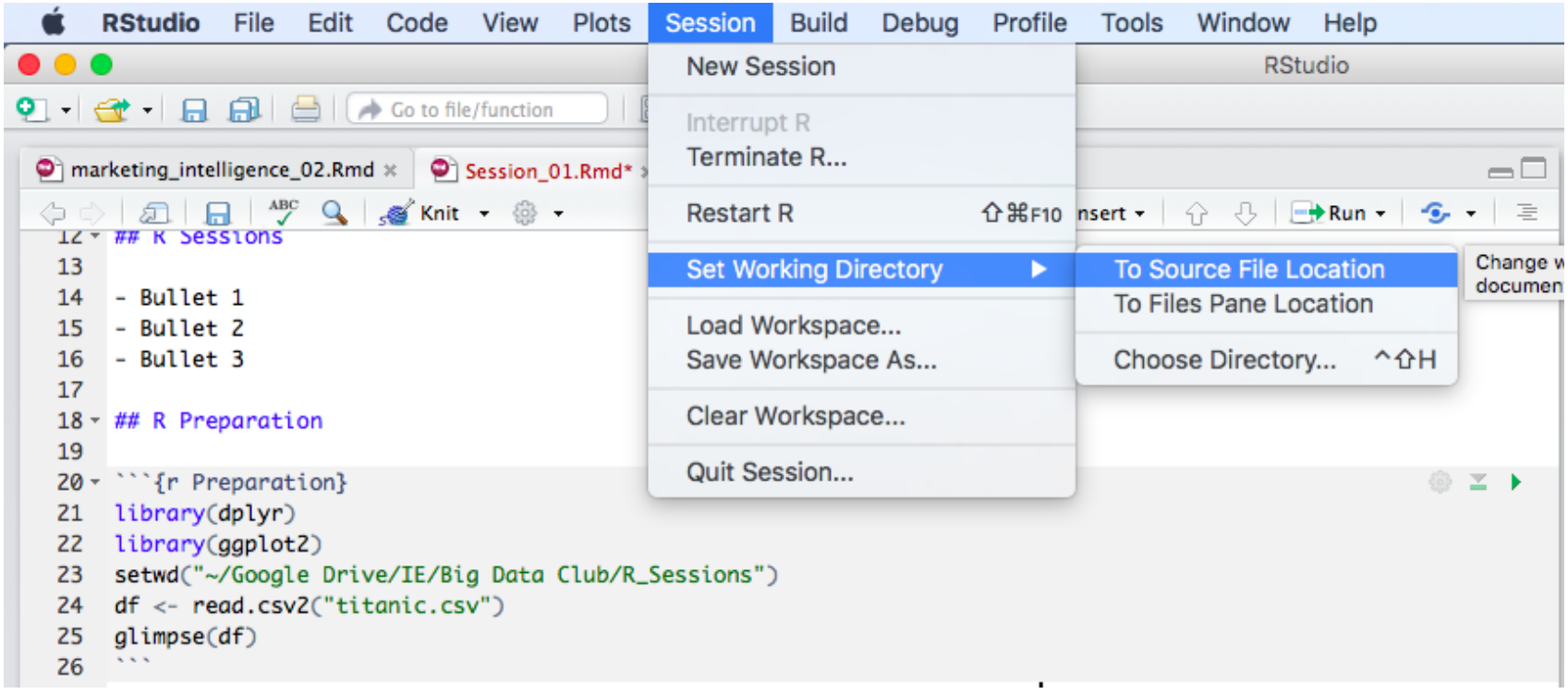

Figure 1:

Preparation

```
library(dplyr)
library(ggplot2)
setwd("~/Google Drive/IE/Big Data Club/R_Sessions")
df <- read.csv2("titanic.csv")
print(str(df))
```

```
## 'data.frame':    1310 obs. of  10 variables:
##  $ pclass  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: int  1 1 0 0 0 1 1 0 1 0 ...
##  $ sex     : Factor w/ 3 levels "","female","male": 2 3
##  $ age     : num  29 0.917 2 30 25 ...
##  $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
##  $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
##  $ ticket  : Factor w/ 930 levels "","110152","110413",
##  $ fare    : num  211 152 152 152 152 ...
##  $ cabin   : Factor w/ 187 levels "","A10","A11",..: 45
##  $ embarked: Factor w/ 4 levels "","C","Q","S": 4 4 4 4
## NULL
```

Simple data transformations - transform data type

```
print(class(df$survived))
```

```
## [1] "integer"
```

```
print(class(df$fare))
```

```
## [1] "numeric"
```

```
df <- as.data.frame(df)
df$survived <- as.factor(df$survived)
df$fare <- as.integer(df$fare)
print(class(df$survived))
```

```
## [1] "factor"
```

```
print(class(df$fare))
```

```
## [1] "integer"
```

Simple data transformations - filter

```
print(any(is.na(df$sex)))
```

```
## [1] FALSE
```

```
print(any(df$sex == ""))
```

```
## [1] TRUE
```

```
print(dim(df))
```

```
## [1] 1310   10
```

```
#dplyr notation
df <- df %>% filter(!is.na(sex))
df <- df %>% filter(sex != "")
print(dim(df))
```

```
## [1] 1309   10
```

Simple data transformations - change values

```
print(df$age[1:10])
```

```
##  [1] 29.0000  0.9167  2.0000 30.0000 25.0000 48.0000 63.
##  [9] 53.0000 71.0000
```

```
df <- df %>% mutate(age = if_else(age < 1.0,0.0,age))
df$age <- as.integer(df$age)
print(df$age[1:10])
```

```
##  [1] 29  0  2 30 25 48 63 39 53 71
```

```
print(class(df$age))
```

```
## [1] "integer"
```

Simple data transformations - select columns

```
names(df)
```

```
##  [1] "pclass"   "survived" "sex"      "age"      "sibsp"
##  [7] "ticket"   "fare"     "cabin"    "embarked"
```

```
df_small <- df %>% select(pclass,survived,sex)
#the same
df_small2 <- df %>% select(pclass:sex)
#the same
df_small3 <- df %>% select(pclass:age,-age)
#the same
df_small4 <- df %>% select(one_of("pclass","survived","sex"
print(names(df_small))
```

```
## [1] "pclass"   "survived" "sex"
```

Functions - in general

```
fun <- function (parameter1, parameter2 = 'default'){
  print(paste("parameter 1 is:",parameter1))
  return_value <- parameter1 + parameter2
  return(return_value)
  #the same: list(return_value,return_value2,return_valueX)
}
value = fun(2,3)
```

```
## [1] "parameter 1 is: 2"
```

```
print(paste("The returned value is: ",value))
```

```
## [1] "The returned value is:  5"
```

Functions - example: Split data

```
splitdf <- function(dataframe, seed=NULL, percentage=0.8) {
  if (!is.null(seed)) set.seed(seed)
  index <- 1:nrow(dataframe)
  numTrainingSamples <- round(length(index) * percentage)
  trainindex <- sample(index, numTrainingSamples)
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset=trainset,testset=testset)
}
split <- splitdf(df)
train <- split$trainset
test <- split$testset
print(paste("nrow of df: ",nrow(df)))
```

```
## [1] "nrow of df:  1309"
```

```
print(paste("nrow of train",nrow(train),"; nrow of test",nr
```

Apply simple model - error fixing

```
> decision_tree <- tree(survived ~.,data=train)
Error in tree(survived ~ ., data = train) :
  factor predictors must have at most 32 levels
> str(train)
'data.frame':   1047 obs. of  10 variables:
 $ pclass  : int  3 3 1 3 3 3 3 3 2 2 ...
 $ survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 1 1 ...
 $ sex     : Factor w/ 3 levels "","female","male": 3 3 2 3 3 2 3 2 3 3 ...
 $ age     : int  NA 20 40 3 33 43 NA NA 26 29 ...
 $ sibsp   : int  0 1 0 4 0 1 0 8 0 1 ...
 $ parch   : int  0 1 0 2 0 6 0 2 0 0 ...
 $ ticket  : Factor w/ 930 levels "","110152","110413",..: 618 249 807 456 526 777 583 780 342 870 ...
 $ fare    : int  6 15 153 31 7 46 7 69 10 27 ...
 $ cabin   : Factor w/ 187 levels "","A10","A11",..: 1 1 74 1 1 1 1 1 1 1 ...
 $ embarked: Factor w/ 4 levels "","C","Q","S": 3 2 4 4 2 4 4 4 4 2 ...
```

Figure 2:

Apply simple model

```
#install.packages("tree") --> if not installed
library(tree)
#random seed
set.seed(27)
df$ticket <- as.integer(df$ticket)
df$cabin <- as.integer(df$cabin)
split <- splitdf(df)
train <- split$trainset
test <- split$testset
decision_tree <- tree(survived ~.,data=train)
prediction <- predict(decision_tree,newdata=test)
print(head(prediction))
```

```
##                0         1
## 1   0.06043956 0.9395604
## 5   0.06043956 0.9395604
## 9   0.06043956 0.9395604
## 10  0.71304348 0.2869565
```

Apply simple model - predict

```r
#random seed
set.seed(27)
prediction <- predict(decision_tree,newdata=test)
threshold = 0.5
prediction_class <- if_else(prediction[,2]>threshold,1,0)
tbl <- table(prediction[,2]>threshold,test$survived)
print(tbl)
```

```
##
##             0   1
##   FALSE 145  51
##   TRUE    7  59
```

```r
print(paste("The accuracy of our model is:",(tbl[1,1]+tbl[2
```

```
## [1] "The accuracy of our model is: 0.778625954198473"
```

Tuning by ROC

```r
#install.packages("pROC") --> if not installed yet
library(pROC)
#random seed
set.seed(27)
prediction <- predict(decision_tree,newdata=test)
predictions_df <- data.frame(survived=test$survived, predic
myROC <- roc(survived ~ prediction, predictions_df)
#choose the best threshold
threshold <- coords(myROC, "best", ret = "threshold")
prediction_class <- if_else(prediction[,2]>threshold,1,0)
tbl <- table(prediction[,2]>threshold,test$survived)
print(tbl)
```

```
##
##             0   1
##   FALSE 110  18
##   TRUE   42  92
```