Data Science Using Python and R

Chantal Larose and Daniel Larose

Chapter 3
Data Preparation

Data Mining and Predictive Analytics, By Daniel Larose and Chantal Larose
John Wiley & Sons, Inc, Hoboken, NJ, 2015.

Bank Marketing Data Set

- bank_marketing_training (26,874 records) and bank_marketing_test (10,255 records) data sets adapted from the UCI Machine Learning Repository bank marketing data set
  - https://archive.ics.uci.edu/ml/datasets/bank+marketing
- 4 predictors or independent variables
  - age, educations, previous_outcome, days_since_previous
- 1 target or dependent variable
  - response
- Phone-based marketing campaign conducted by a Portugal bank to solicit potential customers to subscribe to a term deposit account

Problem Understanding Phase - Objectives

- Learn about current and potential customers' characteristics
- Develop model(s) that will identify likely positive responders
  - Predict the profit from using model(s)

Translate Objectives into a Data Science Problem

- Learn about customers
  - Exploratory data analysis to express simple graphic relationships
    - Histogram of age overlain with information about response yes/no to visualize whether age has a bearing on customer response
  - Cluster to determine if natural groupings occur, such as younger/more-educated vs older/less-educated, and compare to customer response
  - Use Association Rules to find useful relationships, such as "if cell phone, then response = yes" with high support and confidence

Translate Objectives into a Data Science Problem (cont'd)

- Develop a suite of models to identify likely positive responders
  - since response is yes/no (nominal data), we can use classification models but not estimation models
- Potential classification model algorithms
  - Decision Trees
  - Random Forests
  - Naïve Bayes Classification
  - Neural Networks
  - Logistic Regression
- Evaluate each model, such as by using misclassification costs
- Construct a table of the best models and their costs
- Consult with management regarding the best model or models to use during the deployment phase

Data Preparation Phase

- Data are cleaned and prepped for analysis
  - Adding an index field
  - Changing misleading field values
  - Re-expressing categorical data as numeric data
  - Standardizing the numeric fields
  - Identifying outliers

Adding an Index Field

- Augment the data set with new variables to enhance understanding
- For example, add an id field to the data so that if it is resorted, you can return back to the original order of the data

In Python

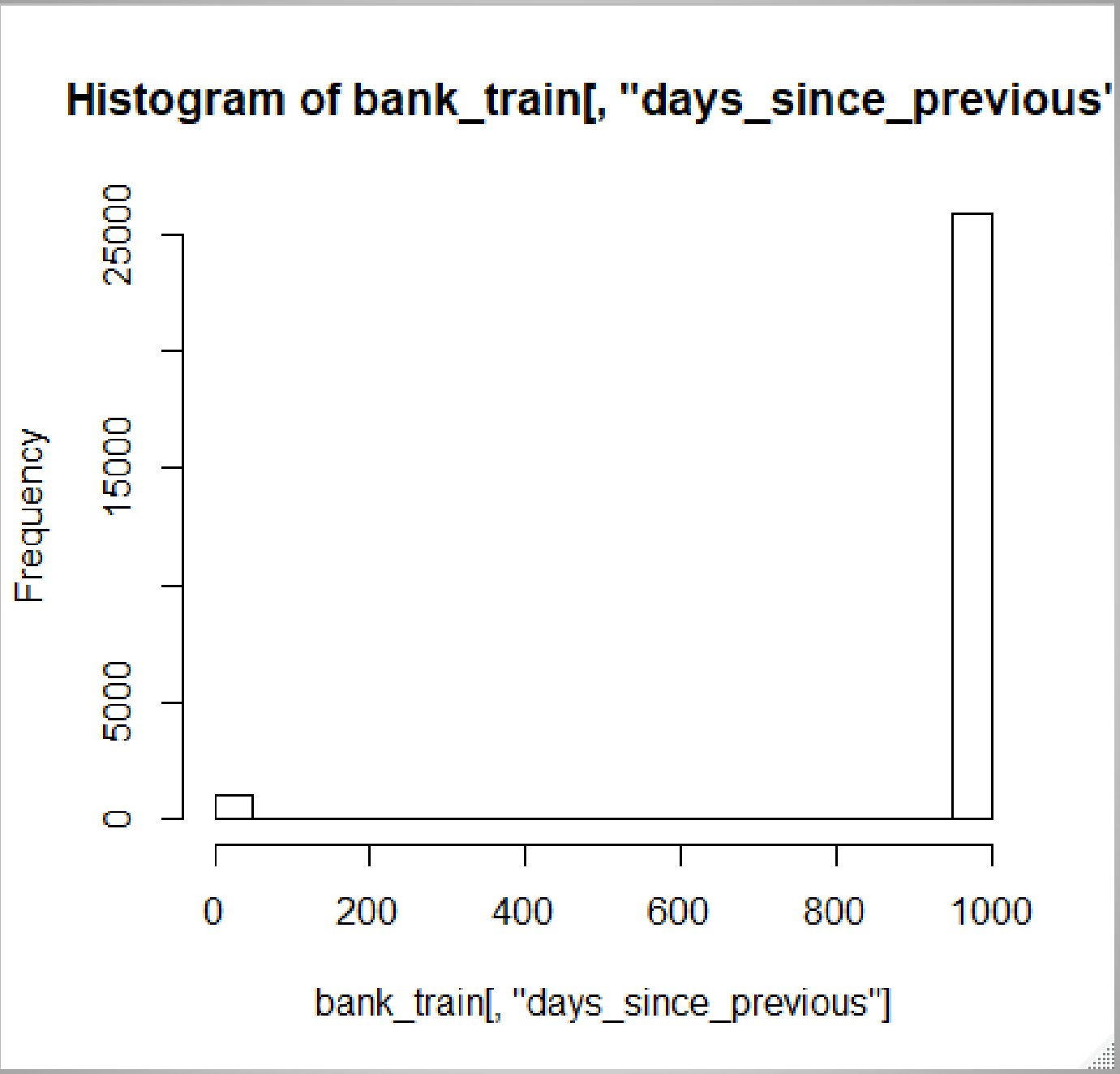
- import pandas as pd
- bank_train = pd.read_csv("bank_marketing_training")
- print('bank_train.shape is', bank_train.shape)
- bank_train['index'] = pd.Series(range(0,26874))
- print(bank_train.head)

In R

- bank_train <-read.csv("bank_marketing_training")
- n <- dim(bank_train)[1]
- bank_train$Index <- c(1:n)
- head(bank_train)

Changing Misleading Field Values

- Some attributes may have values that are codes or meant to signify that values are missing
- For example, days_since_previous in the bank data records the number of days since the client was contacted during a previous campaign
- We can use R to show a histogram of the values of this field
  - hist(bank_train[,"days_since_previous"])


Histogram of bank_train[, "days_since_previous'
Frequency
0
5000
15000
25000
0
200
400
600
800
1000
bank_train[, "days_since_previous"]

Changing Misleading Field Values (cont'd)

- We note that a few values are near zero with a majority of the values near 1000
- The database administrator used the code 999 to represent customers who had not been contacted previously
- Need to change 999 to missing

In Python

- import numpy as np
- bank_train['days_since_previous'] = bank_train['days_since_previous'].replace({999: np.NaN})
- bank_train['days_since_previous'].plot(kind='hist',title='Histogram of Days Since Previous')

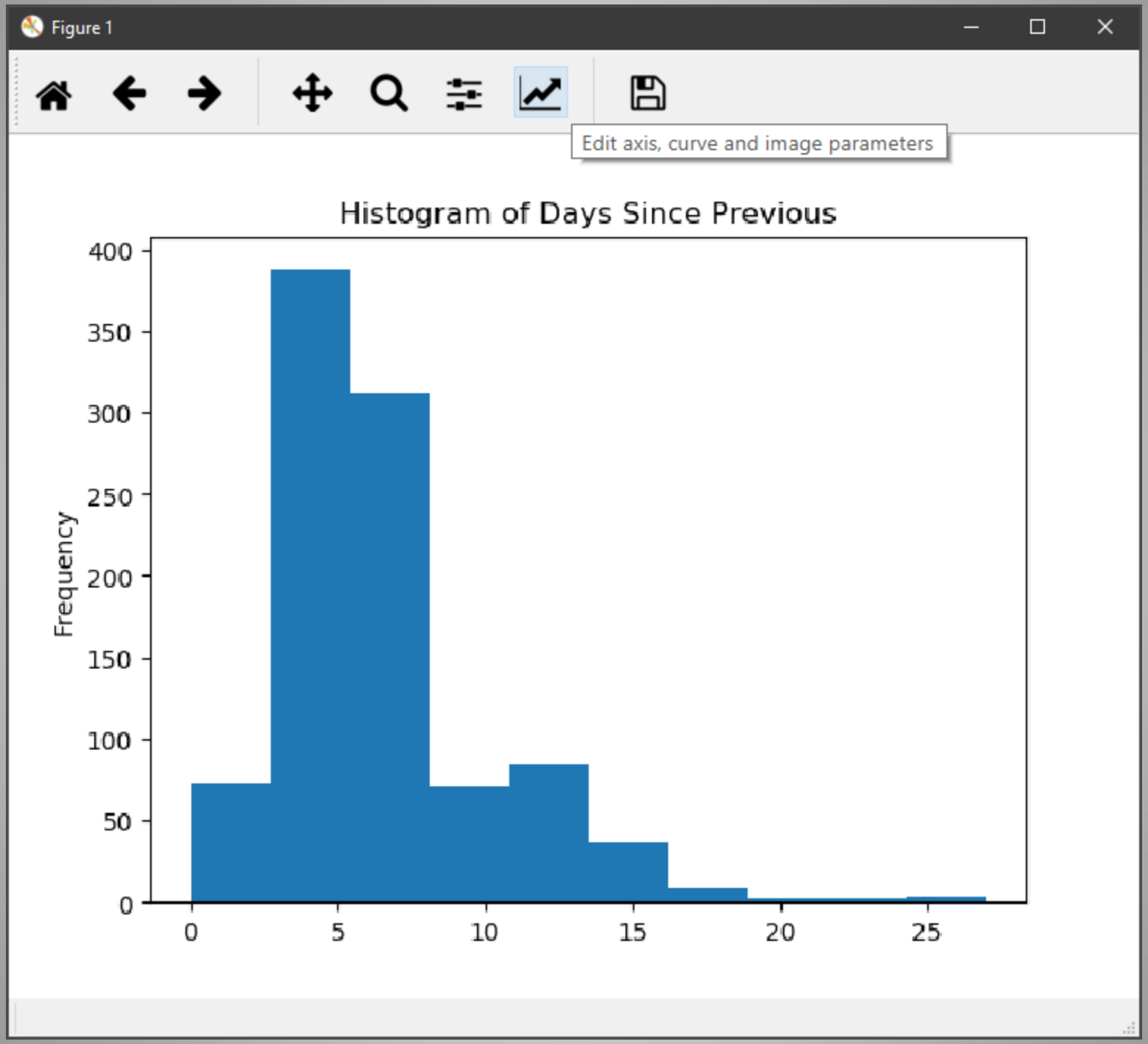
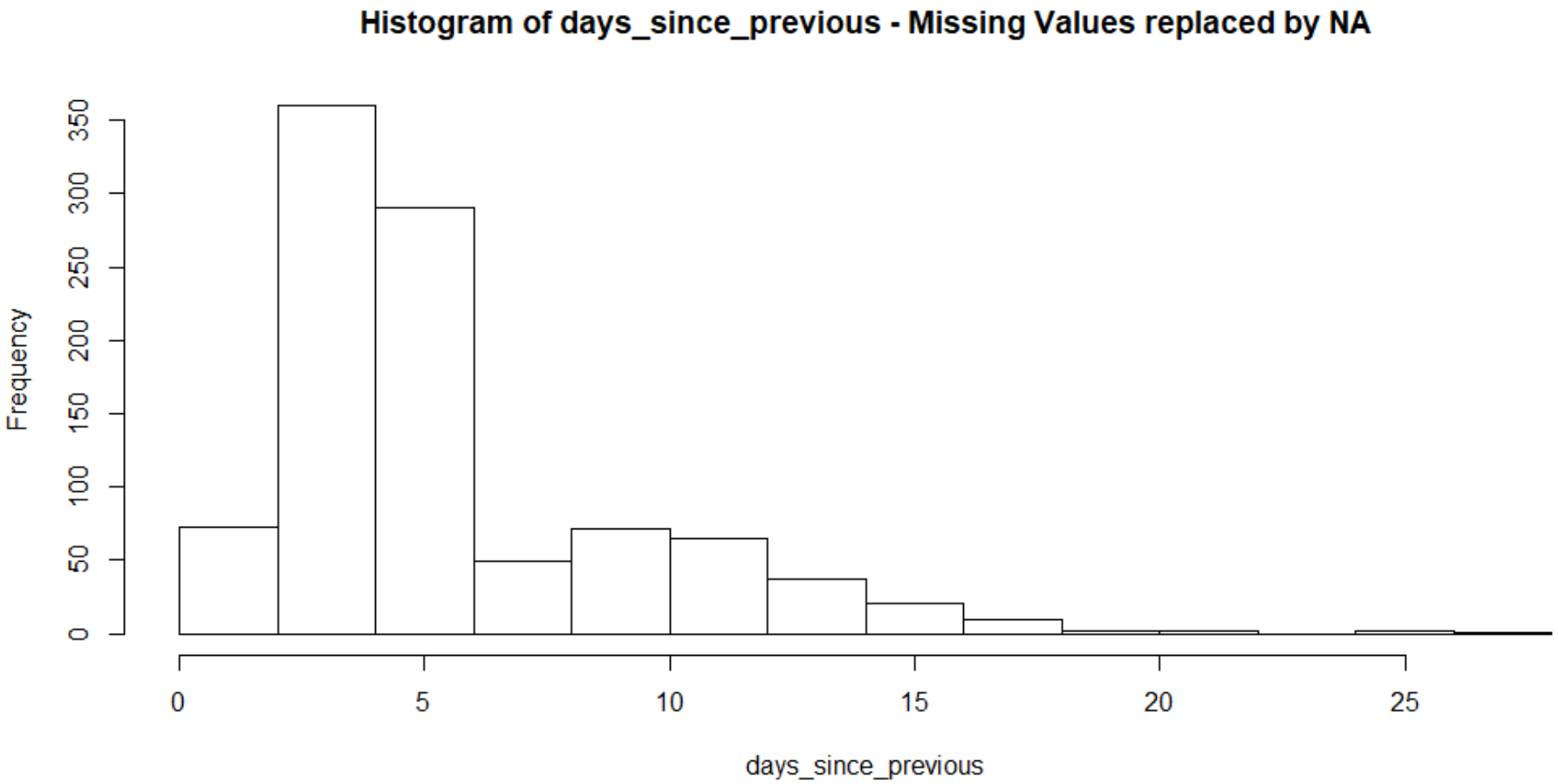


Figure 1
Edit axis, curve and image parameters
Histogram of Days Since Previous
Frequency
400
350
300
250
200
150
100
50
0
0
5
10
15
20
25

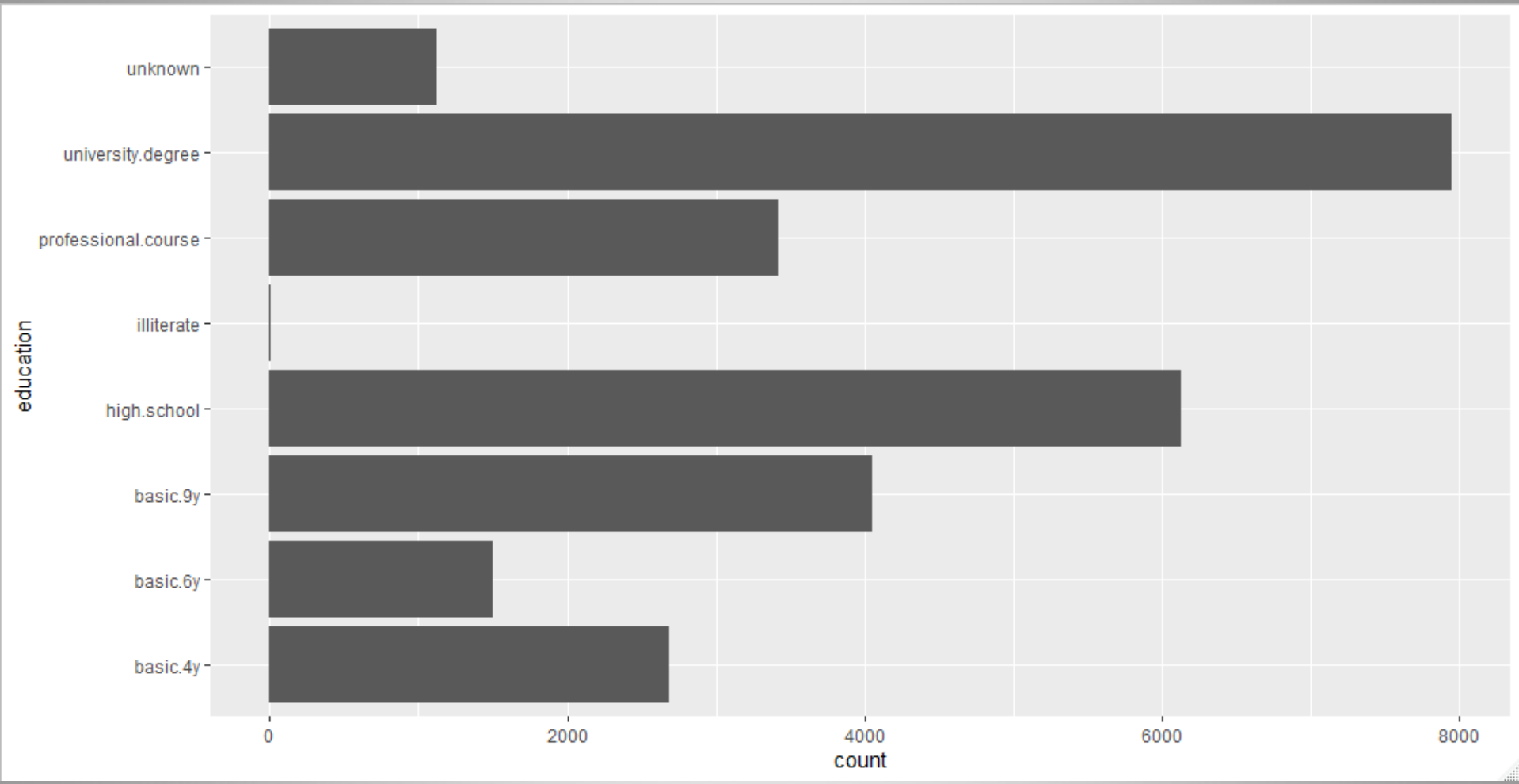
In R

- hist(bank_train[,"days_since_previous"])
- bank_train$days_since_previous <-
- ifelse(test=bank_train$days_since_previous == 999,
- yes = NA, no = bank_train$days_since_previous)
- hist(bank_train$days_since_previous,xlab="days_since_previous", main="Histogram of days_since_previous - Missing Values replaced by NA")


Histogram of days_since_previous - Missing Values replaced by NA
Frequency
0
50
100
150
200
250
300
350
0
5
10
15
20
25
days_since_previous

Re-Expression of Categorical Data as Numeric

- On the next slide is shown a bar chart of the education field in the bank data
- The data is categorical and would not let the machine algorithms know that there is a difference in the education levels
- We need to proceed carefully in changing the categorical values to numbers


unknown
university.degree
professional.course
illiterate
high.school
basic.9y
basic.6y
basic.4y
education
0
2000
4000
6000
8000
count

Proposed Numeric Values

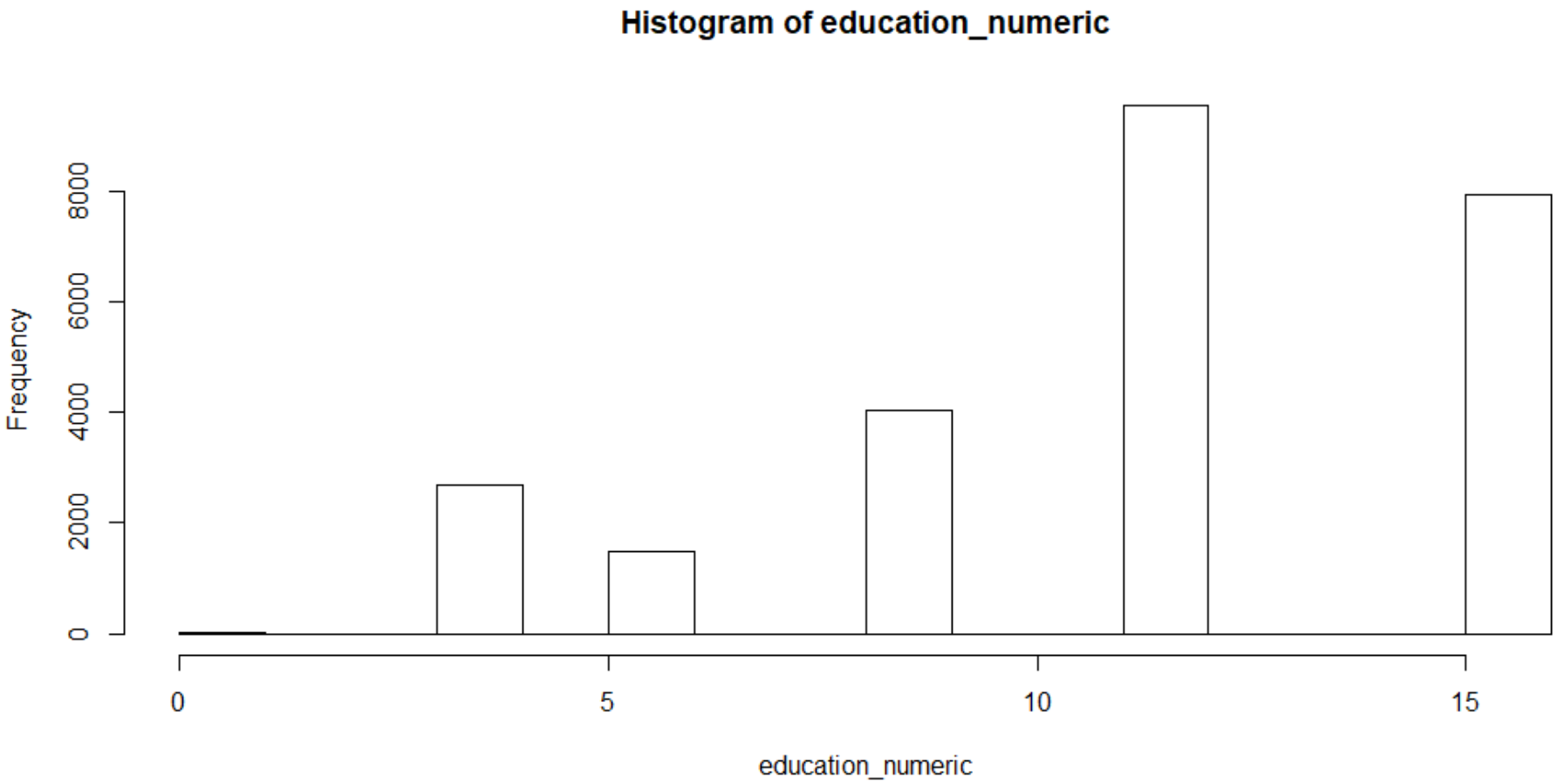
- illiterate – 0
- basic.4y – 4
- basic.6y – 6
- basic.9y – 9
- high.school – 12
- professional.course – 12
  - Taken from http://invet-project.eu/wp-content/uploads/2014/06/National-Report_Portugal_Final.pdf
- university.degree – 16
- unknown - missing

In Python

- bank_train['education_numeric'] = bank_train['education']
- dict_edu = {"education_numeric":{"illiterate":0,"basic.4y":4,"basic.6y":6,"basic.9y":9,"high.school":12,"professional.course":12,"university.degree":16,"unknown":np.NaN}}
- bank_train.replace(dict_edu,inplace=True)
- print(bank_train['education_numeric'])

In R

- library(plyr)
- edu.num <- revalue(x=bank_train$education,replace=c("illiterate"=0,"basic.4y"=4,"basic.6y"=6, "basic.9y"=9, "high.school"=12,"professional.course"=12, "university.degree"=16,"unknown"=NA))
- bank_train$education_numeric <- as.numeric(levels(edu.num))[edu.num]
- hist(bank_train$education_numeric,xlab="education_numeric",main="Histogram of education_numeric")


Histogram of education_numeric
Frequency
0
2000
4000
6000
8000
0
5
10
15
education_numeric

Standardizing the Numeric Fields

- Certain algorithms perform better when standardizing the field means to equal 0 and the field standard deviation equals 1
- Example: Z-Value
  - z = Standardized Value = $(x-\overline{x})/s$ = (data value – mean)/(standard deviation)
  - Positive z-values may be interpreted as representing the number of standard deviations above the mean while negative z-values would be the number of standard deviations below the mean

In Python

- from scipy import stats
- bank_train['age_z'] = stats.zscore(bank_train['age'])
- print(bank_train['age_z'])

In R

- bank_train$age_z <- scale(x=bank_train$age)

Identifying Outliers

- Z-values can be used to identify outliers which are records of extreme values along a particular dimension or dimensions
- Example: number_of_contacts in the bank data which represent the number of customer contacts made over the marketing campaign
  - Mean: 2.6
  - Standard Deviation: 2.7
  - number_of_contacts_z = (number_of_contacts-2.6)/2.7
- A data value may be considered to be an outlier if its z-value is either greater than 3 or less than -3
  - Example: number_of_contacts_z = (10-2.6)/2.7 = 2.7, so, 10 would not be an outlier even though 10 contacts seems like a lot
- Outliers should only be removed if approved by the client

In Python

- bank_train_outliers = bank_train.query('age_z > 3 | age_z < -3')
- bank_train_sort = bank_train.sort_values(['age_z'],ascending=False)
- #age and marital status of the 15 people with the largest age_z values
- print(bank_train_sort[['age','marital']].head(n=15))

In R

- bank_outliers <- bank_train[which(bank_train$age_z <- -3 | bank_train$age_z > 3),]
- bank_train_sort <- bank_train[order(-bank_train$age_z),]
- bank_train_sort[1:15,]
- head(bank_train_sort)
- bank_train_sort[1:15,c(1,3)]

Saving the Updated Data

- Both Python and R support writing a dataframe to a csv file

In Python

- bank_train.to_csv('new_bank_train',index=False)

In R

- write.csv(bank_train,"new_bank_train",row.names=FALSE)