Data Science Using Python and R

Chantal Larose and Daniel Larose

Chapter 4
Data Exploration

Hypotheses

- Clients may have a priori hypotheses to test, such as
  - Do cellphone users have a higher rate of positive responses than landline users?
  - Testing may be done with
    - Classical statistical methods
    - Cross-validation methods of data science
- If no a priori hypotheses to test, then use exploratory or graphical data analysis

Exploratory/Graphical Data Analysis

- Use graphics to explore the relationship between predictor and target variables
- Use graphics and tables to derive new variables to increase predictive value
- Use binning productively, to increase predictive value

Bar Graphs with Response Overlay

- Bar graphs with a response overlay show the relationship between the target variable and a categorical predictor
- In addition, we can normalize the bar graph to equalize the length of each bar to more easily see the response portions

In Python

- import pandas as pd
- bank_train = pd.read_csv("bank_marketing_training")
- crosstab_01 = pd.crosstab(bank_train['previous_outcome'], bank_train['response'])
- crosstab_01.plot(kind='bar', stacked=True)
- #div divides each row of crosstab_01 (axis=0) by the row sum (row(1))
- crosstab_norm = crosstab_01.div(crosstab_01.sum(1),axis=0)
- crosstab_norm.plot(kind='bar', stacked = True)

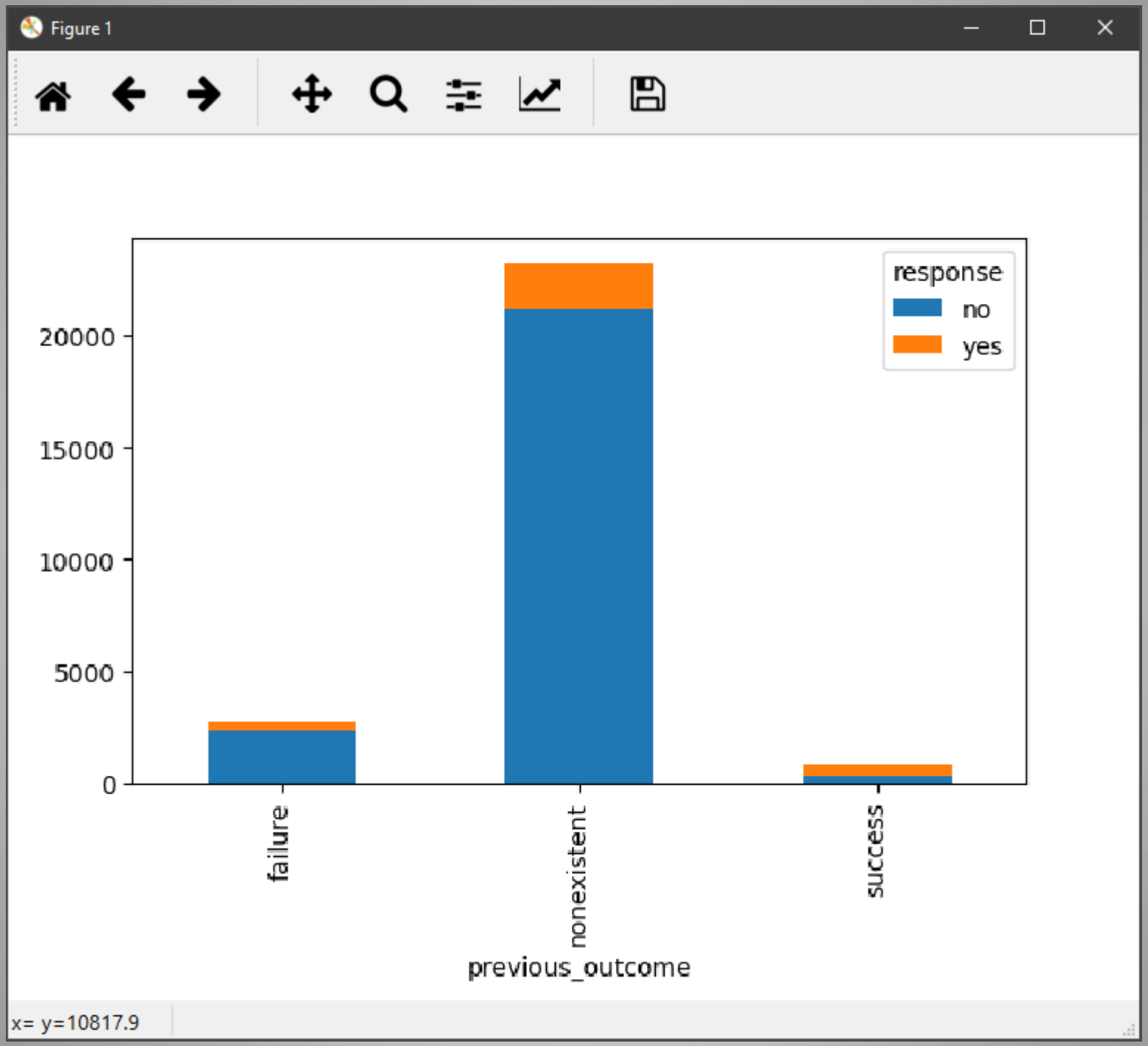


Figure 1
response
no
yes
20000
15000
10000
5000
0
failure
nonexistent
success
previous_outcome
x= y=10817.9

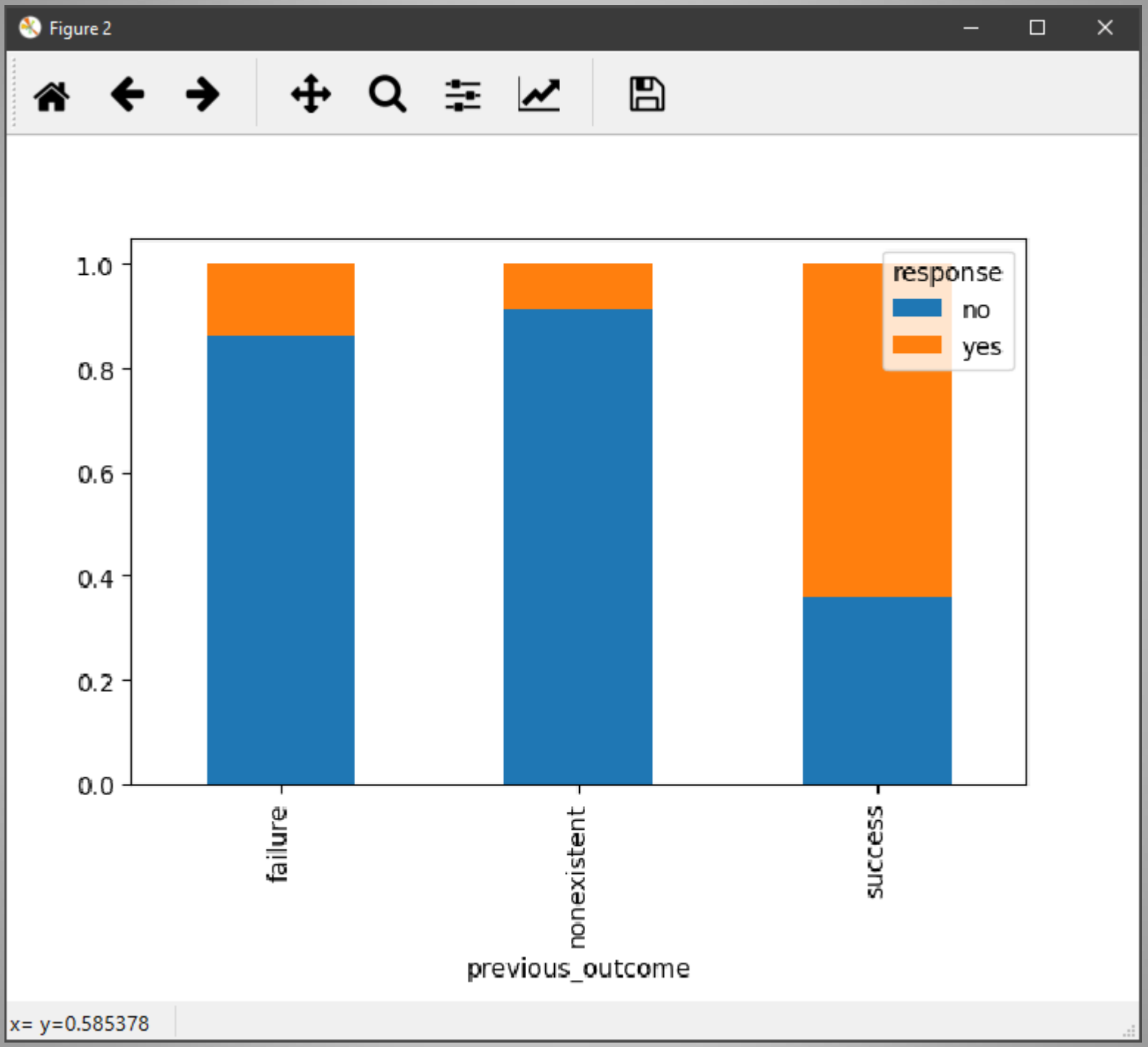
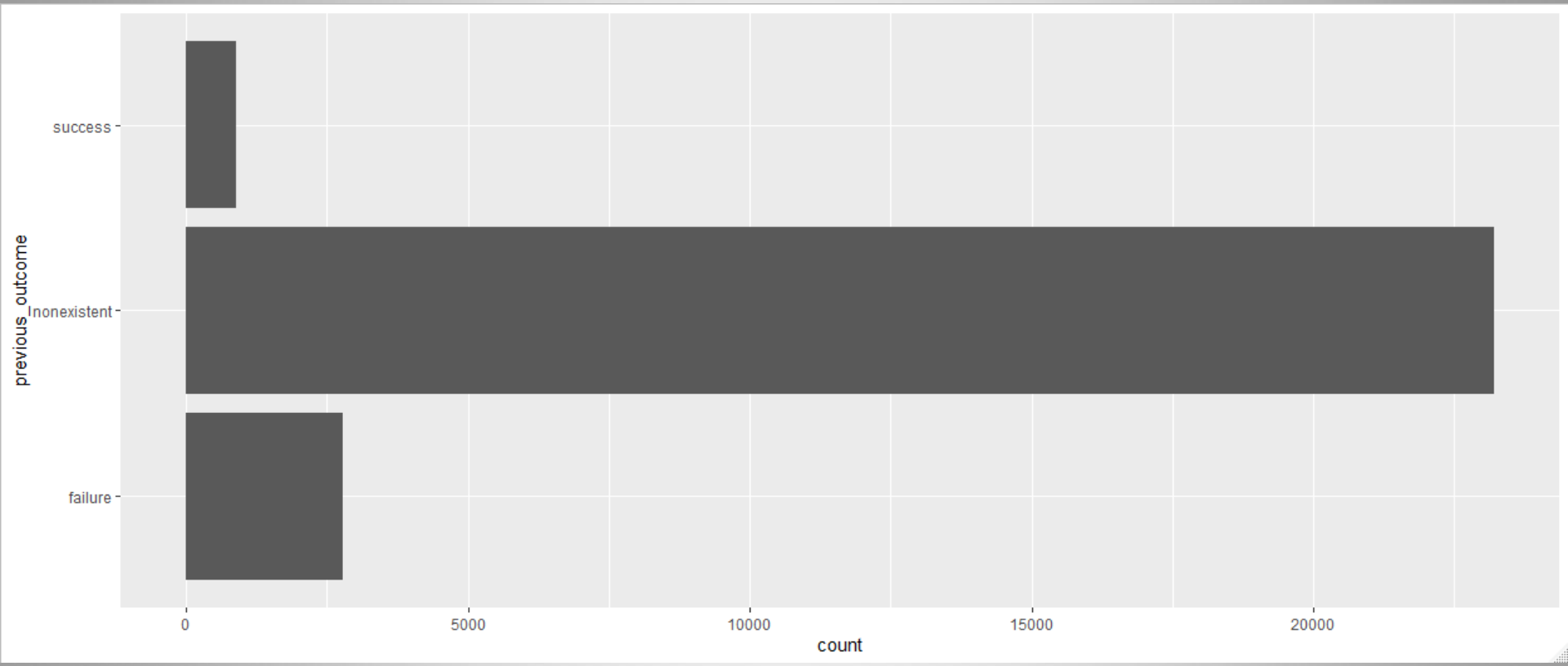
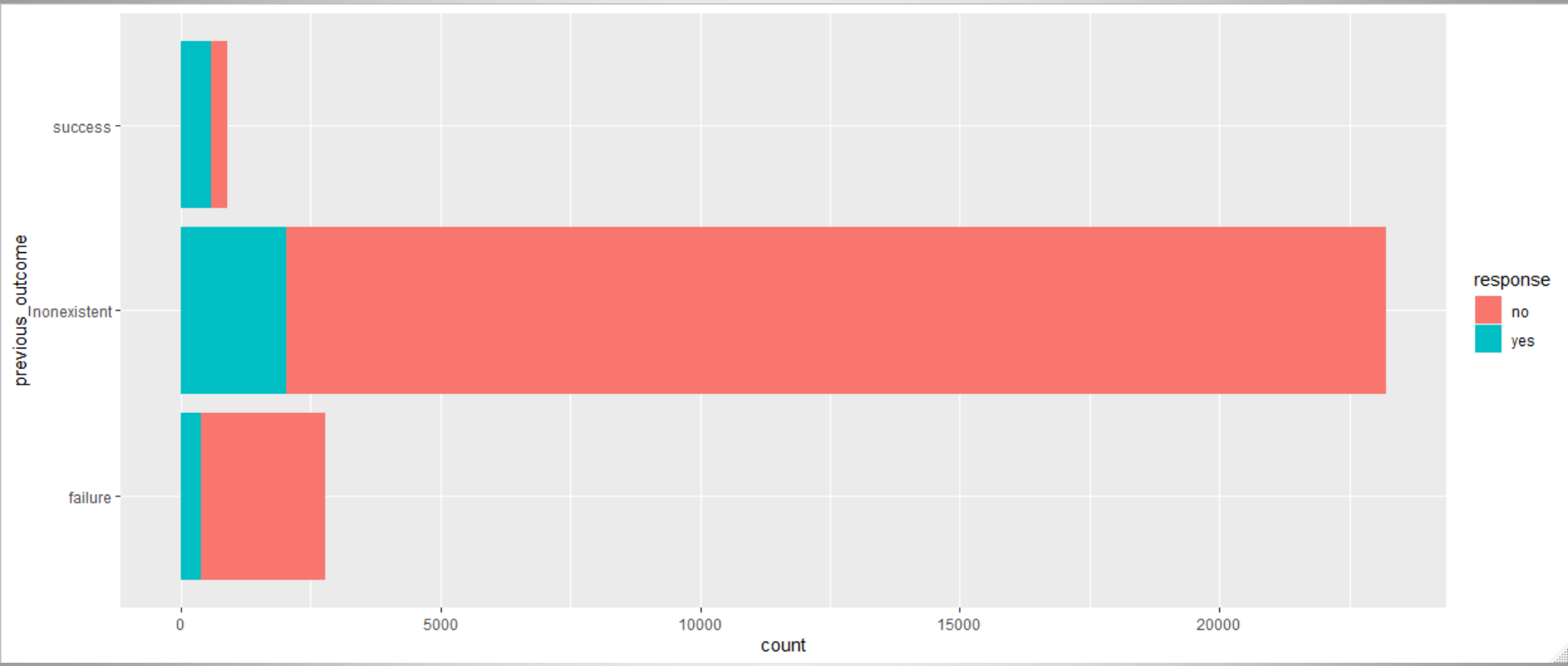


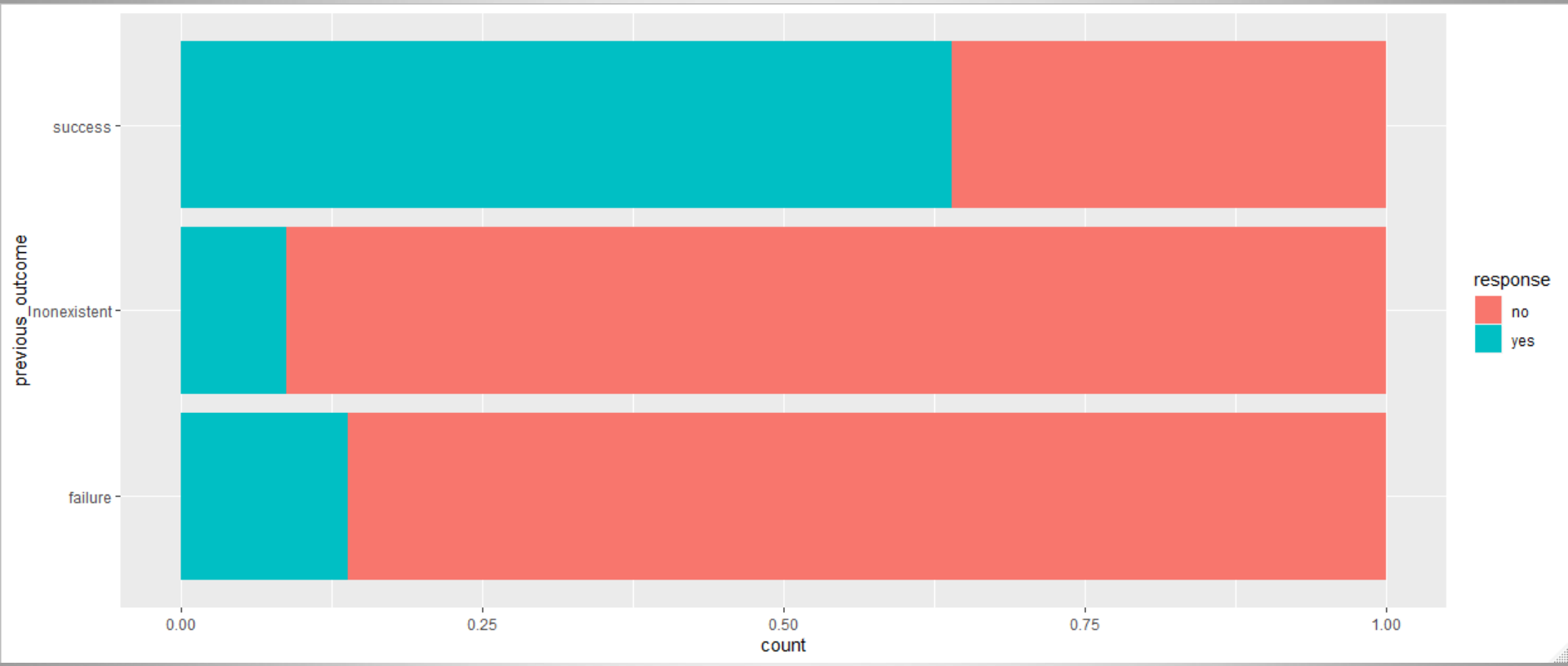
Figure 2
1.0
0.8
0.6
0.4
0.2
0.0
response
no
yes
failure
nonexistent
success
previous_outcome
x= y=0.585378

In R

- library(ggplot2)
- bank_train <- read.csv("bank_marketing_training")
- #aes - aesthetics
- ggplot(bank_train,aes(previous_outcome)) + geom_bar() + coord_flip()
- #response overlay
- ggplot(bank_train, aes(previous_outcome)) + geom_bar(aes(fill=response)) + coord_flip()
- #normalize
- ggplot(bank_train, aes(previous_outcome)) + geom_bar(aes(fill=response), position="fill") + coord_flip()


success
nonexistent
failure
previous_outcome
0
5000
10000
15000
20000
count


success
nonexistent
failure
previous_outcome
0
5000
10000
15000
20000
count
response
no
yes


success
nonexistent
failure
previous_outcome
0.00
0.25
0.50
0.75
1.00
count
response
no
yes

Contingency Tables

- Contingency tables help to examine the relationship between 2 variables, such as a predictor variable and a target variable
  - Usual practice is to have target variable along the rows and the predictor variable along the columns
  - We can examine the actual number and the percentages

In Python

- #change over response to rows and previous_outcome to columns
- crosstab_02 = pd.crosstab(bank_train['response'],bank_train['previous_outcome'])
- #round rounds to a specified number of digits, in this case 1
- crosstab_norm_02 = round(crosstab_02.div(crosstab_02.sum(0),axis=1)*100,1)

```
crosstab_02
Out[10]:
previous_outcome  failure  nonexistent  success
response
no                2390     21176        320
yes               385      2034         569

crosstab_norm_02
Out[11]:
previous_outcome  failure  nonexistent  success
response
no                86.1     91.2         36.0
yes               13.9     8.8          64.0
```

In R

- t.v1 <- table(bank_train$response,bank_train$previous_outcome)
- t.v2 <- addmargins(A=t.v1,FUN=list(total=sum),quiet=TRUE)
- t.v1.rnd <- round(prop.table(t.v1,margin=2)*100,1)

```
> t.v1

       failure nonexistent success
  no      2390        21176     320
  yes      385         2034     569
> t.v2

         failure nonexistent success total
  no        2390        21176     320 23886
  yes        385         2034     569  2988
  total     2775        23210     889 26874
> t.v1.rnd

       failure nonexistent success
  no      86.1        91.2    36.0
  yes     13.9         8.8    64.0
>
```

Histograms with Response Overlay

- Histograms are a graphical representation of a frequency distribution for a numerical variable
  - They also may be normalized to see the response patterns

In Python

- import numpy as np
- import matplotlib.pyplot as plt
- bt_age_y = bank_train[bank_train.response == "yes"]['age']
- bt_age_n = bank_train[bank_train.response == "no"]['age']
- #stacked histogram
- plt.figure() #new plot
- plt.hist([bt_age_y,bt_age_n], bins=10, stacked=True)
- plt.legend(['Response=Yes','Response=No'])
- plt.title('Histogram of Age with Response Overlay')
- plt.xlabel('Age'); plt.ylabel('Frequency'); plt.show()

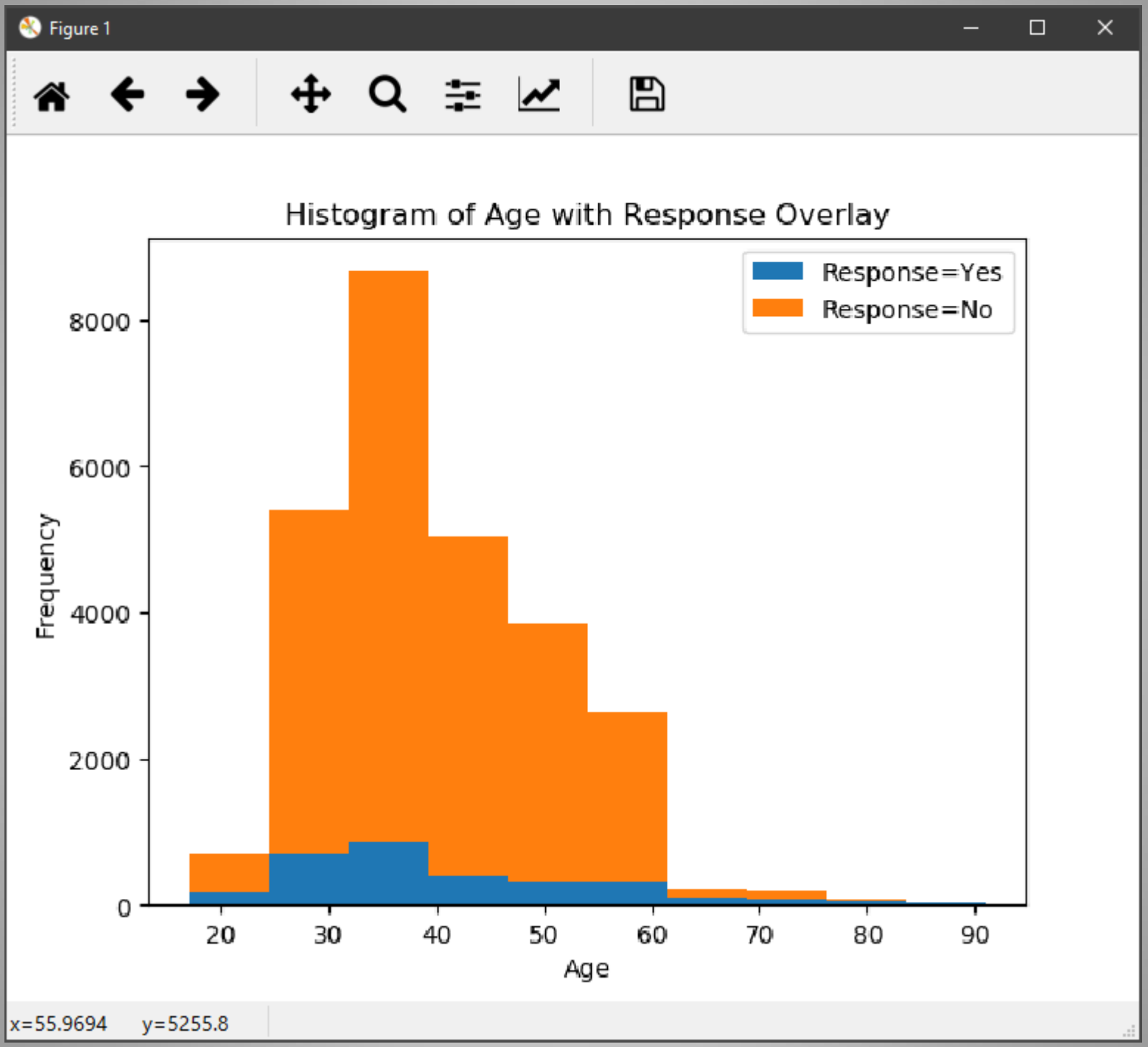


Figure 1
Histogram of Age with Response Overlay
Response=Yes
Response=No
Frequency
8000
6000
4000
2000
0
20
30
40
50
60
70
80
90
Age
x=55.9694
y=5255.8

In Python

- #normalized
- (n,bins,patches) = plt.hist([bt_age_y,bt_age_n], bins=10, stacked=True)
- #create a new plot
- plt.figure()
- n_table = np.column_stack((n[0],n[1]))
- n_norm = n_table / n_table.sum(axis=1)[:,None]
- #create an array of bin cuts
- ourbins = np.column_stack((bins[0:10],bins[1:11]))
- p1=plt.bar(x=ourbins[:,0],height=n_norm[:,0],width=ourbins[:,1]-ourbins[:,0])
- p2=plt.bar(x=ourbins[:,0],height=n_norm[:,1],width=ourbins[:,1]-ourbins[:,0],bottom=n_norm[:,0])
- plt.legend(['Response=Yes','Response=No'])
- plt.title('Normalized Histogram of Age with Response Overlay')
- plt.xlabel('Age'); plt.ylabel('Proportion'); plt.show()

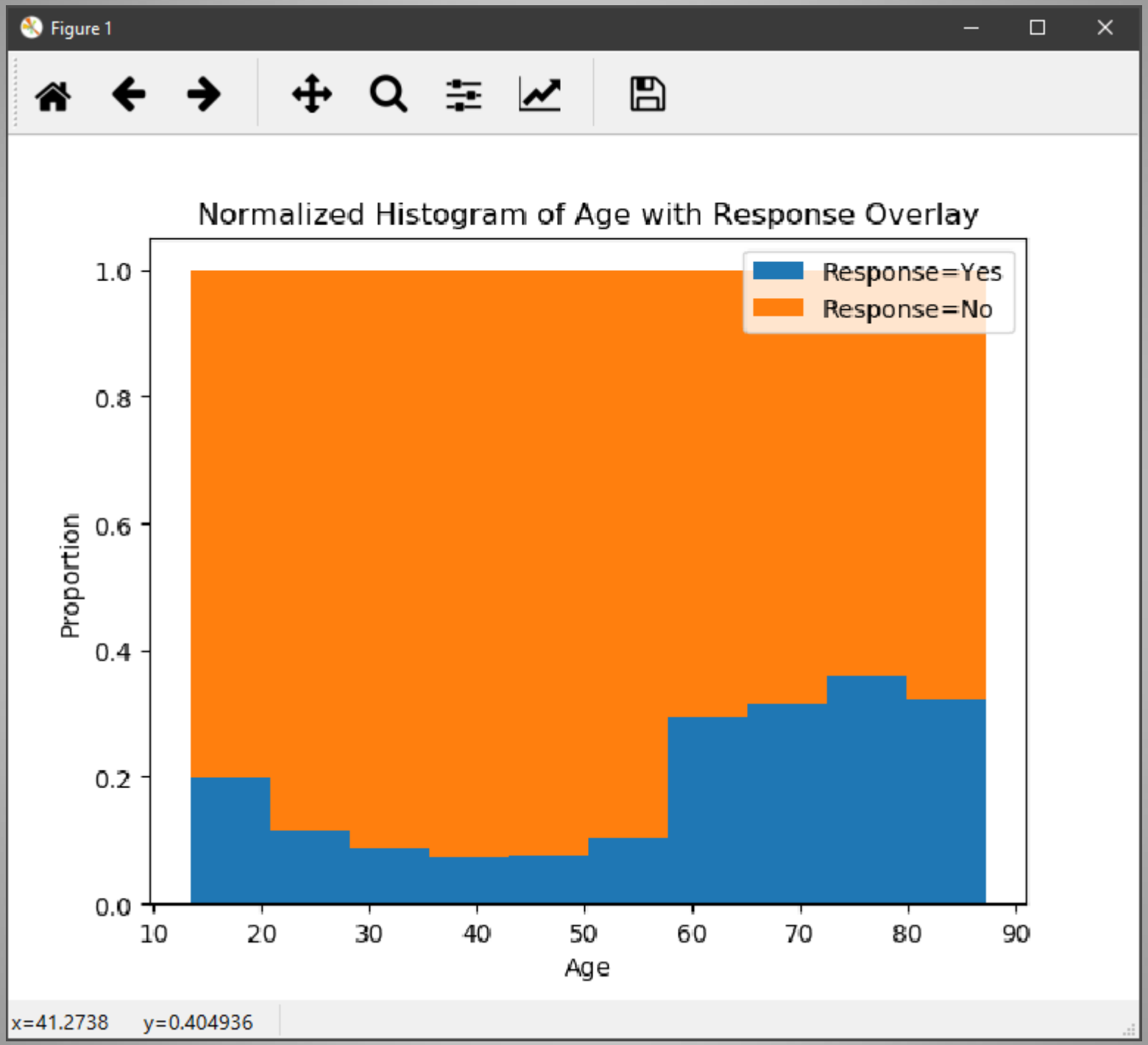
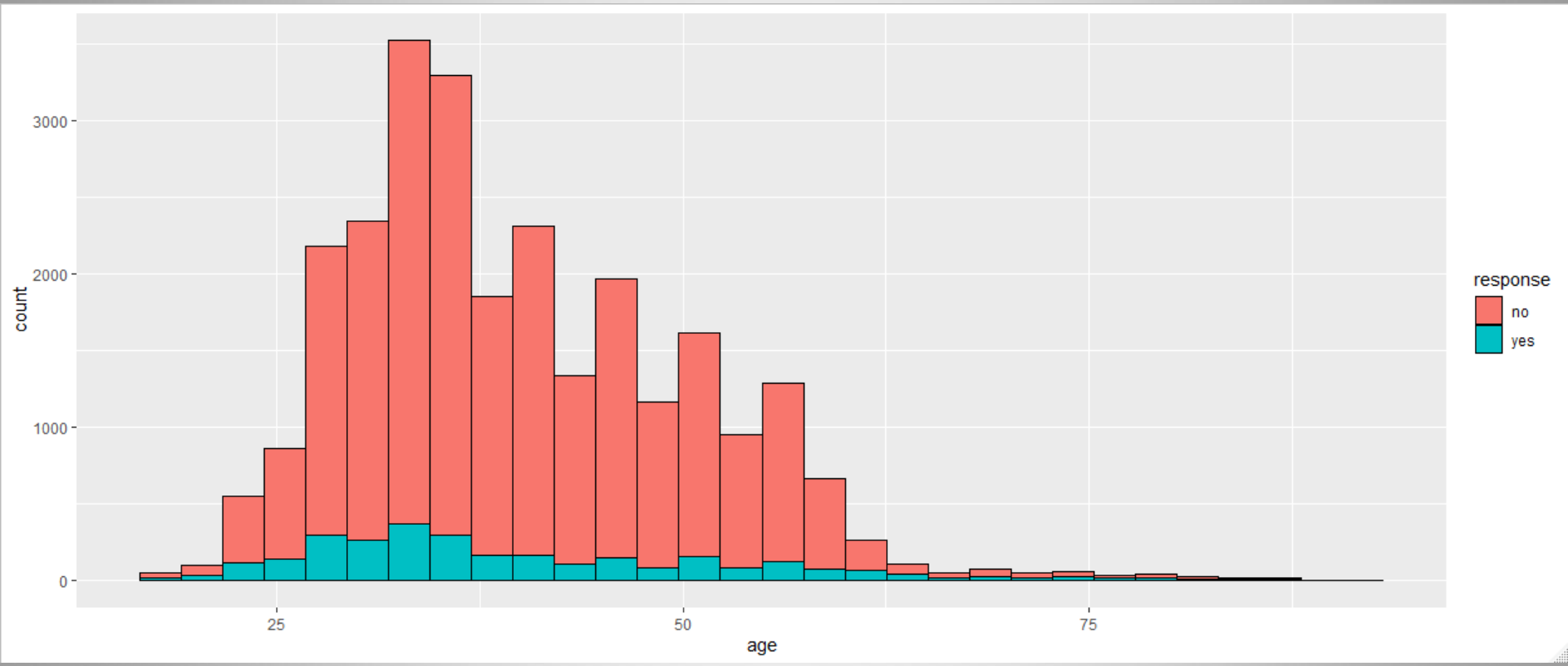
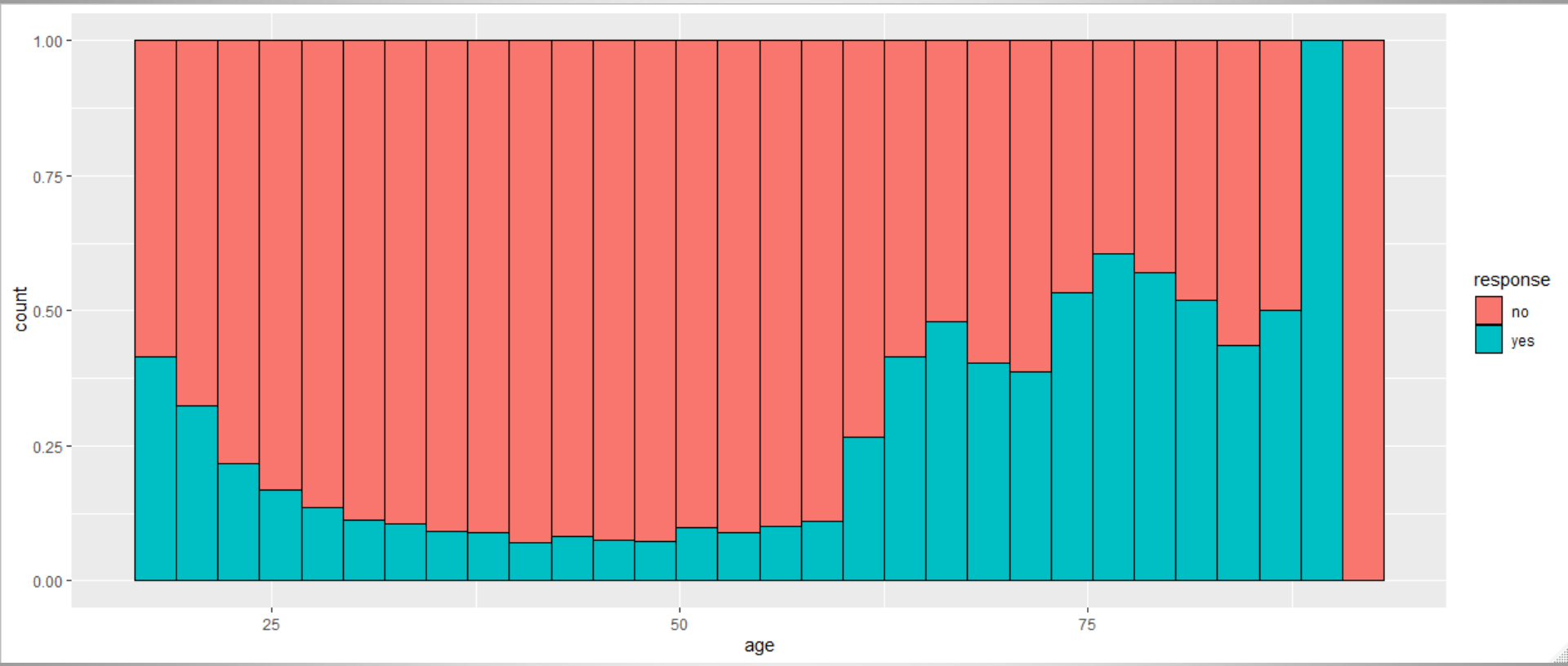


Figure 1
Normalized Histogram of Age with Response Overlay
Response=Yes
Response=No
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
10
20
30
40
50
60
70
80
90
Age
x=41.2738
y=0.404936

In R

- library(ggplot2)
- bank_train <- read.csv("bank_marketing_training")
- #histograms
- ggplot(bank_train, aes(age)) + geom_histogram(aes(fill=response), color="black")
- #normalized
- ggplot(bank_train, aes(age)) + geom_histogram(aes(fill=response), color="black", position="fill")


3000
2000
1000
0
count
25
50
75
age
response
no
yes


1.00
0.75
0.50
0.25
0.00
count
25
50
75
age
response
no
yes

Binning Based on Predictive Value

- When we see how a numeric predictor variable relates to our target variable, we can use the relationship to convert the numeric predictor variable to a categorical predictor variable to take advantage of more models
- For example, in the graph on the previous slide, we see that the following approximate ranges can be used for age
  - 1: under 27 (moderate response)
  - 2: 27 to 60 (low response, over 90% of customers)
  - 3: above 60 (high response)

In Python

- import pandas as pd
- bank_train = pd.read_csv("bank_marketing_training")
- #right=False means exclude the right hand bin points
- bank_train['age_binned'] = pd.cut(x=bank_train['age'],bins=[0,27,60.01,100],labels=["Under 27","27 to 60", "Over 60"],right=False)
- crosstab_03 = pd.crosstab(bank_train['age_binned'],bank_train['response'])
- crosstab_03.plot(kind='bar',stacked=True,title='Bar Graph of Age (Binned) with Response Overlay')
- #div divides each row of crosstab_03 (axis=0) by the row sum (row(1))
- crosstab_norm_03 = crosstab_03.div(crosstab_03.sum(1),axis=0)
- crosstab_norm_03.plot(kind='bar', stacked = True)

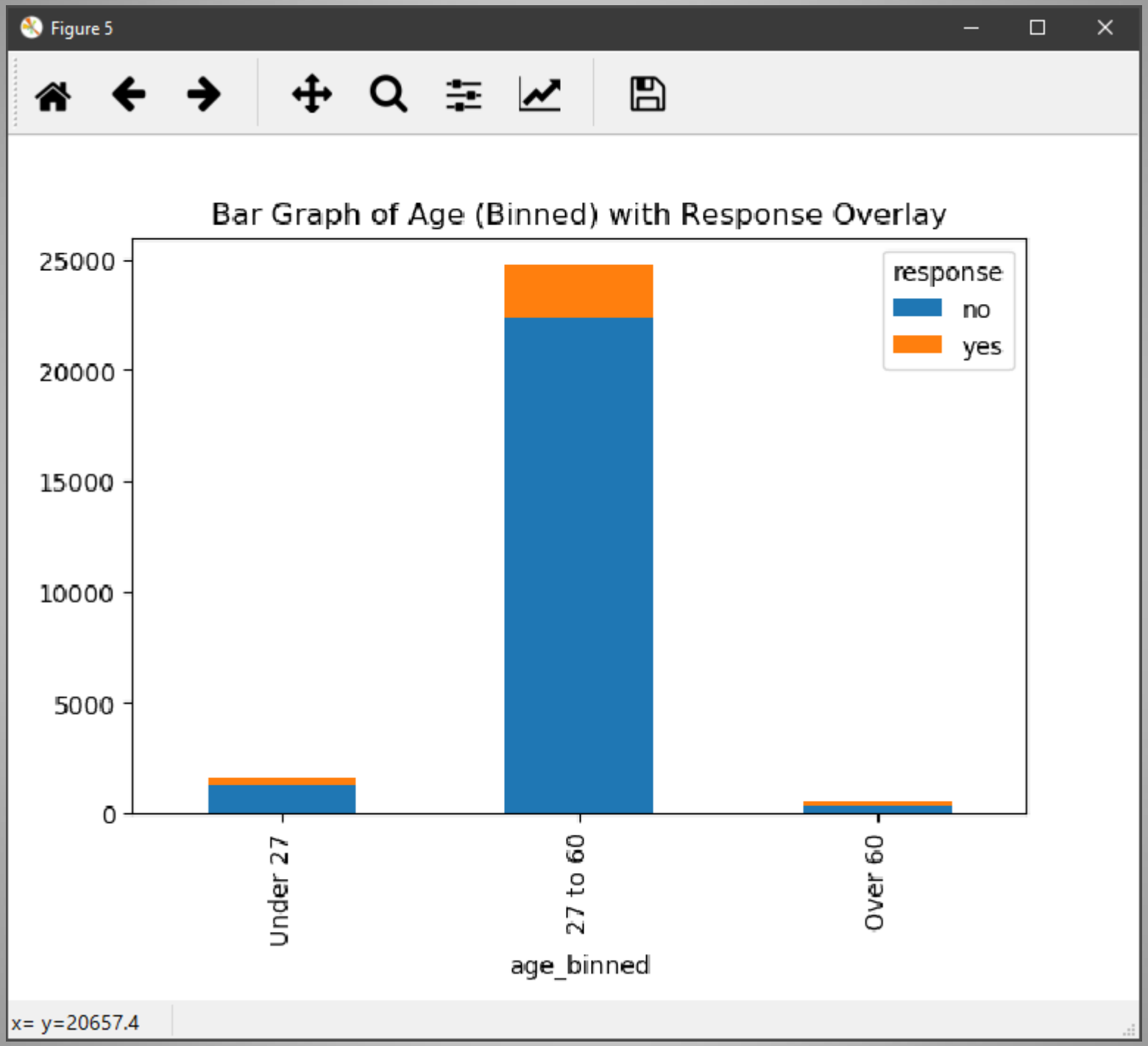


Figure 5
Bar Graph of Age (Binned) with Response Overlay
25000
20000
15000
10000
5000
0
response
no
yes
Under 27
27 to 60
Over 60
age_binned
x= y=20657.4

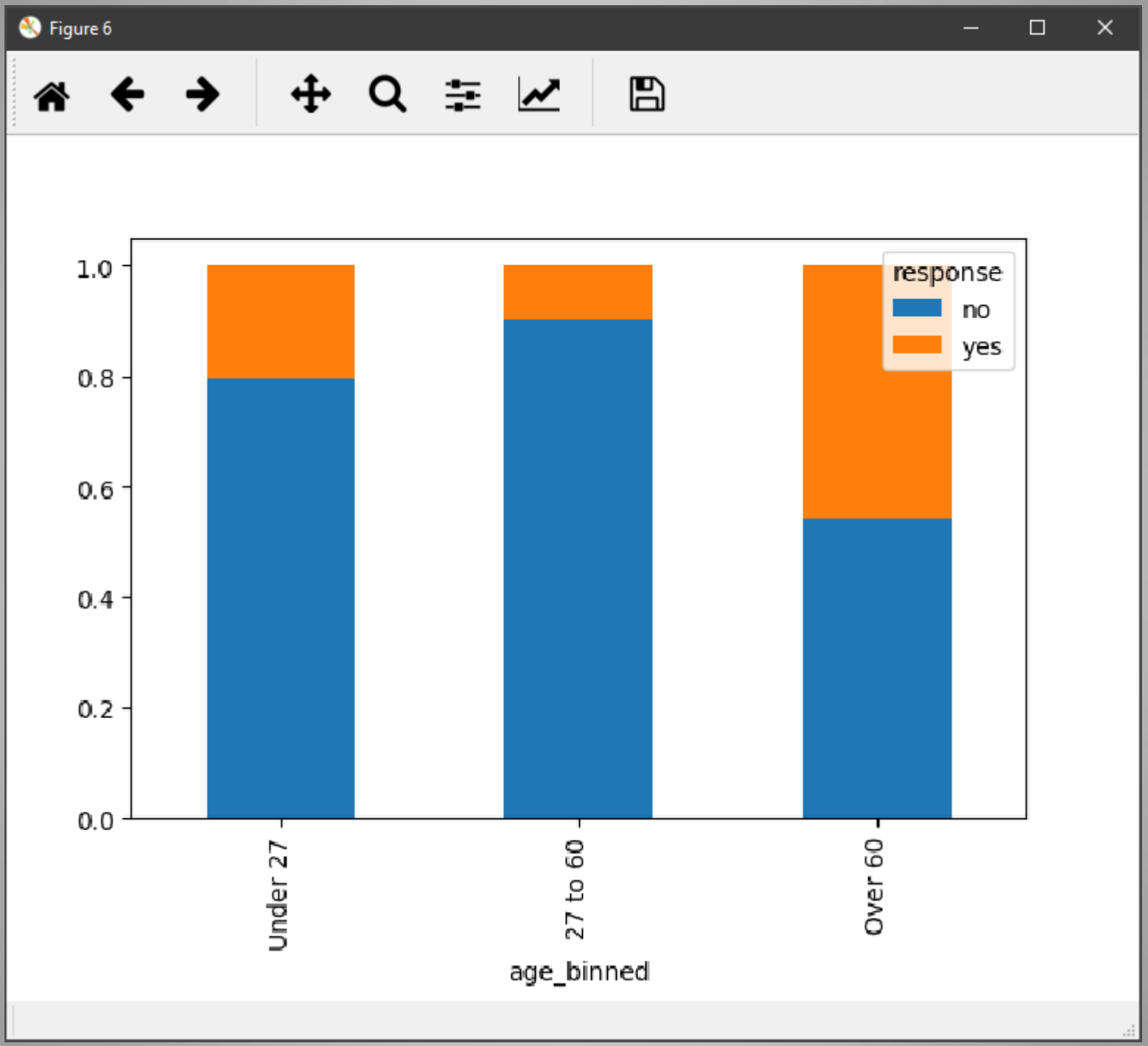
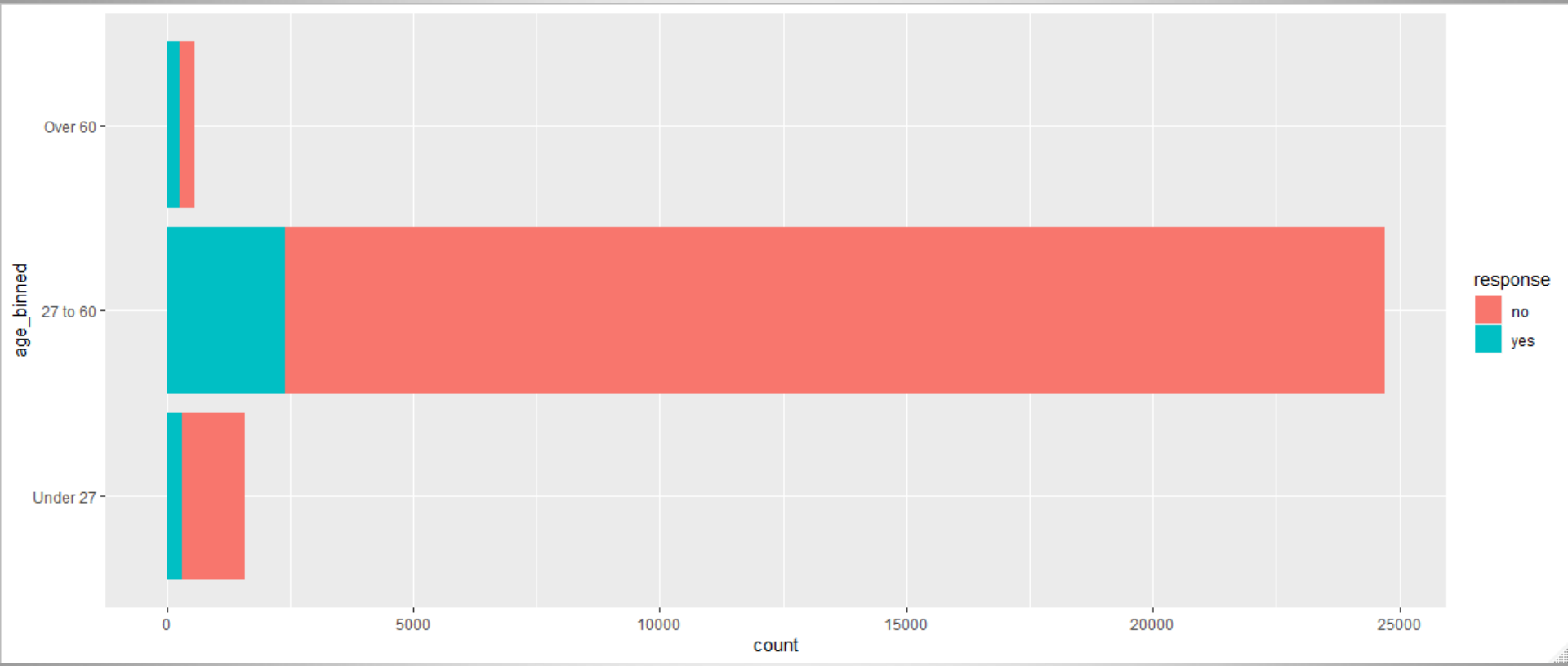
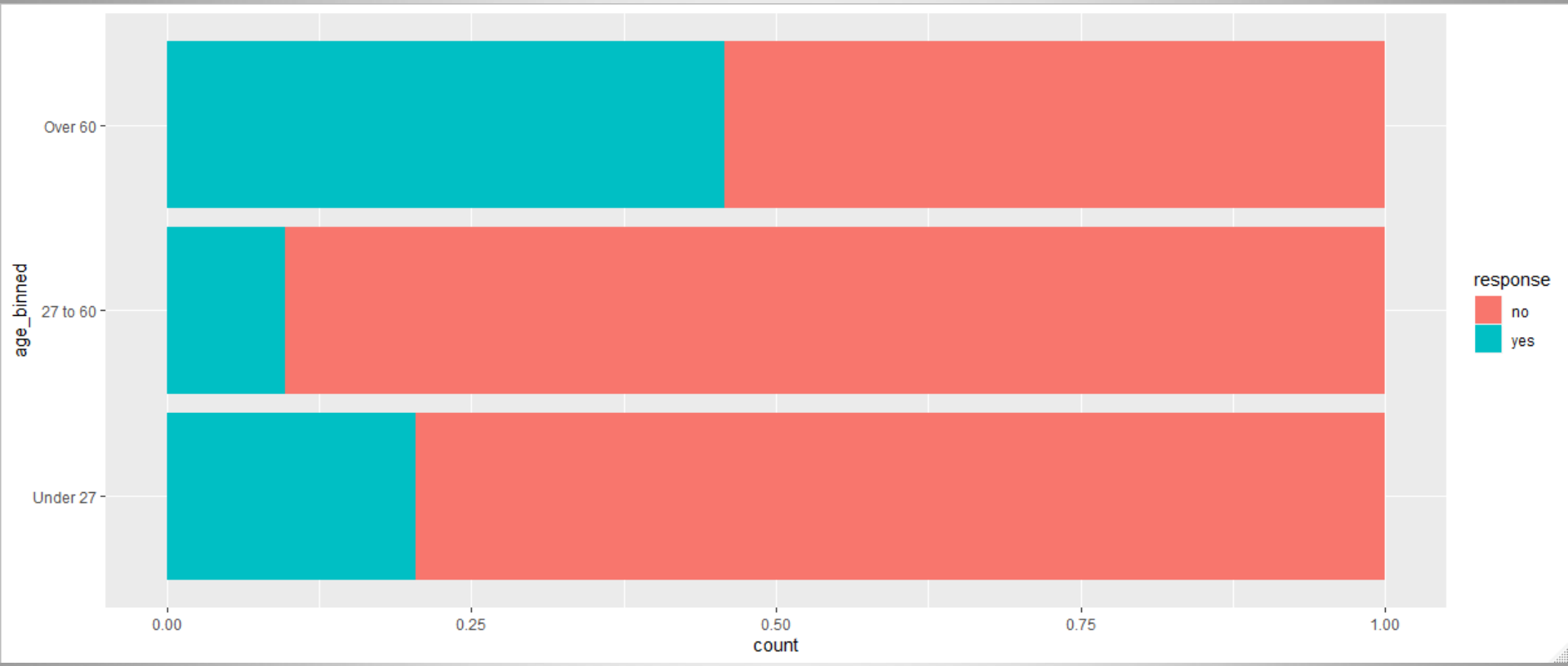


Figure 6
1.0
0.8
0.6
0.4
0.2
0.0
response
no
yes
Under 27
27 to 60
Over 60
age_binned

In R

- library(ggplot2)
- bank_train <- read.csv("bank_marketing_training")
- #Binning
- bank_train$age_binned <-
  cut(x=bank_train$age,breaks=c(0,27,60.01,100),right=FALSE,
  labels=c("Under 27","27 to 60","Over 60"))
- ggplot(bank_train, aes(age_binned)) +
  geom_bar(aes(fill=response)) + coord_flip()
- ggplot(bank_train, aes(age_binned)) +
  geom_bar(aes(fill=response), position="fill") + coord_flip()
- #Contingency Table
- t2.v1 <- table(bank_train$response,bank_train$age_binned)
- t2.v2 <- addmargins(A=t2.v1,FUN=list(total=sum),quiet=TRUE)
- t2.rnd <- round(prop.table(t2,margin=2)*100,1)


Over 60
27 to 60
Under 27
age_binned
0
5000
10000
15000
20000
25000
count
response
no
yes


Over 60
27 to 60
Under 27
age_binned
0.00
0.25
0.50
0.75
1.00
count
response
no
yes

```
~/R/chapter03/
> t2.v1

     Under 27 27 to 60 Over 60
no       1255    22315      316
yes       322     2399      267
> t2.v2

       Under 27 27 to 60 Over 60 total
no         1255    22315      316 23886
yes         322     2399      267  2988
total      1577    24714      583 26874
> t2.rnd

     Under 27 27 to 60 Over 60
no       79.6     90.3    54.2
yes      20.4      9.7    45.8
>
```