Conversion Rate Analysis

Howard Nguyen

—

```
library(dplyr)  # Data Wrangling and Manipulation
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

Load data

```
conversionsDF <- read.csv(file="bank-additional-full.csv", header=TRUE, sep=";")
```

```
# Shape of conversionsDF
dim(conversionsDF)
```

```
## [1] 41188    21
```

```
# Quick look at conversionsDF
head(conversionsDF)
```

```
##   age       job marital   education default housing loan   contact month
## 1  56 housemaid married    basic.4y      no      no   no telephone   may
## 2  57  services married high.school unknown      no   no telephone   may
## 3  37  services married high.school      no     yes   no telephone   may
## 4  40    admin. married    basic.6y      no      no   no telephone   may
## 5  56  services married high.school      no      no  yes telephone   may
## 6  45  services married    basic.9y unknown      no   no telephone   may
##   day_of_week duration campaign pdays previous    poutcome emp.var.rate
## 1         mon      261        1   999        0 nonexistent          1.1
## 2         mon      149        1   999        0 nonexistent          1.1
```

```
## 3         mon      226        1   999        0 nonexistent          1.1
## 4         mon      151        1   999        0 nonexistent          1.1
## 5         mon      307        1   999        0 nonexistent          1.1
## 6         mon      198        1   999        0 nonexistent          1.1
##   cons.price.idx cons.conf.idx euribor3m nr.employed  y
## 1         93.994         -36.4     4.857        5191 no
## 2         93.994         -36.4     4.857        5191 no
## 3         93.994         -36.4     4.857        5191 no
## 4         93.994         -36.4     4.857        5191 no
## 5         93.994         -36.4     4.857        5191 no
## 6         93.994         -36.4     4.857        5191 no
```

the output variable, y, which has information on whether a client has subscribed to a term deposit, is encoded as 'yes' or 'no'. In order to simplify our conversion rate computations, we will encode this variable as 1 for 'yes' and 0 for 'no'.

```
# Encode conversions as 0s and 1s
conversionsDF$y <- ifelse(conversionsDF$y=="yes",1,0)
conversionsDF$conversion <- as.integer(conversionsDF$y)
```

```
tail(conversionsDF)
```

```
##       age         job marital           education default housing loan  contact
## 41183  29  unemployed  single            basic.4y      no     yes   no cellular
## 41184  73     retired married professional.course      no     yes   no cellular
## 41185  46 blue-collar married professional.course      no      no   no cellular
## 41186  56     retired married   university.degree      no     yes   no cellular
## 41187  44  technician married professional.course      no      no   no cellular
## 41188  74     retired married professional.course      no     yes   no cellular
##       month day_of_week duration campaign pdays previous    poutcome
## 41183   nov         fri      112        1     9        1     success
## 41184   nov         fri      334        1   999        0 nonexistent
## 41185   nov         fri      383        1   999        0 nonexistent
## 41186   nov         fri      189        2   999        0 nonexistent
## 41187   nov         fri      442        1   999        0 nonexistent
## 41188   nov         fri      239        3   999        1     failure
##       emp.var.rate cons.price.idx cons.conf.idx euribor3m nr.employed y
## 41183         -1.1         94.767         -50.8     1.028      4963.6 0
## 41184         -1.1         94.767         -50.8     1.028      4963.6 1
## 41185         -1.1         94.767         -50.8     1.028      4963.6 0
## 41186         -1.1         94.767         -50.8     1.028      4963.6 0
## 41187         -1.1         94.767         -50.8     1.028      4963.6 1
## 41188         -1.1         94.767         -50.8     1.028      4963.6 0
##       conversion
## 41183          0
## 41184          1
## 41185          0
## 41186          0
## 41187          1
## 41188          0
```

1. Aggregate Conversion Rate

Since we have already encoded the output variable as 1 for those who have converted and 0 for those who have not in a column, named conversion, we can simply sum over this column to get the total number of conversions. The following code snippet shows how we can sum over the conversion column and get the total number of clients in the data:

```
# total of numbers of conversions
sprintf("total conversions: %i out of %i", sum(conversionsDF$conversion), nrow(conversionsDF))
```

```
## [1] "total conversions: 4640 out of 41188"
```

```
# total number of clients in the data (= number of records in the data)
sprintf("conversion rate: %0.2f%%", sum(conversionsDF$conversion)/nrow(conversionsDF)*100.0)
```

```
## [1] "conversion rate: 11.27%"
```

2. Conversion Rates by Age

Aggregate conversion rate tells us the overall performance of our marketing campaign. However, it does not give us that much insight. When we are reporting and tracking the progress of marketing efforts, we typically would want to dive deeper into the data and break down the customer base into multiple segments and compute KPIs for individual segments. We will first break our data into smaller segments by age and see how the conversion rates differ by different age groups.

Note: The pipe operator, %>%, in this code, is the way you can apply different functions sequentially. In this code snippet, we are passing conversionDF to a group_by function, then passing the results of this group_by function to the summarise function, and lastly to the mutate function.

In the group_by function, we are grouping the DataFrame by the column age. Then, for each age group, we are counting the number of records in each group, by using a function, n(), and naming it TotalCount. Also, we are summing over the column, conversion, for each age group, by using the sum function, and naming it NumConversions.

Lastly, we are using the mutate function, which adds new variables, while preserving the original DataFrame, to compute conversion rates for each age group. As we can see, we are simply dividing NumConversion by TotalCount and multiplying it by 100 to get the conversion rates.

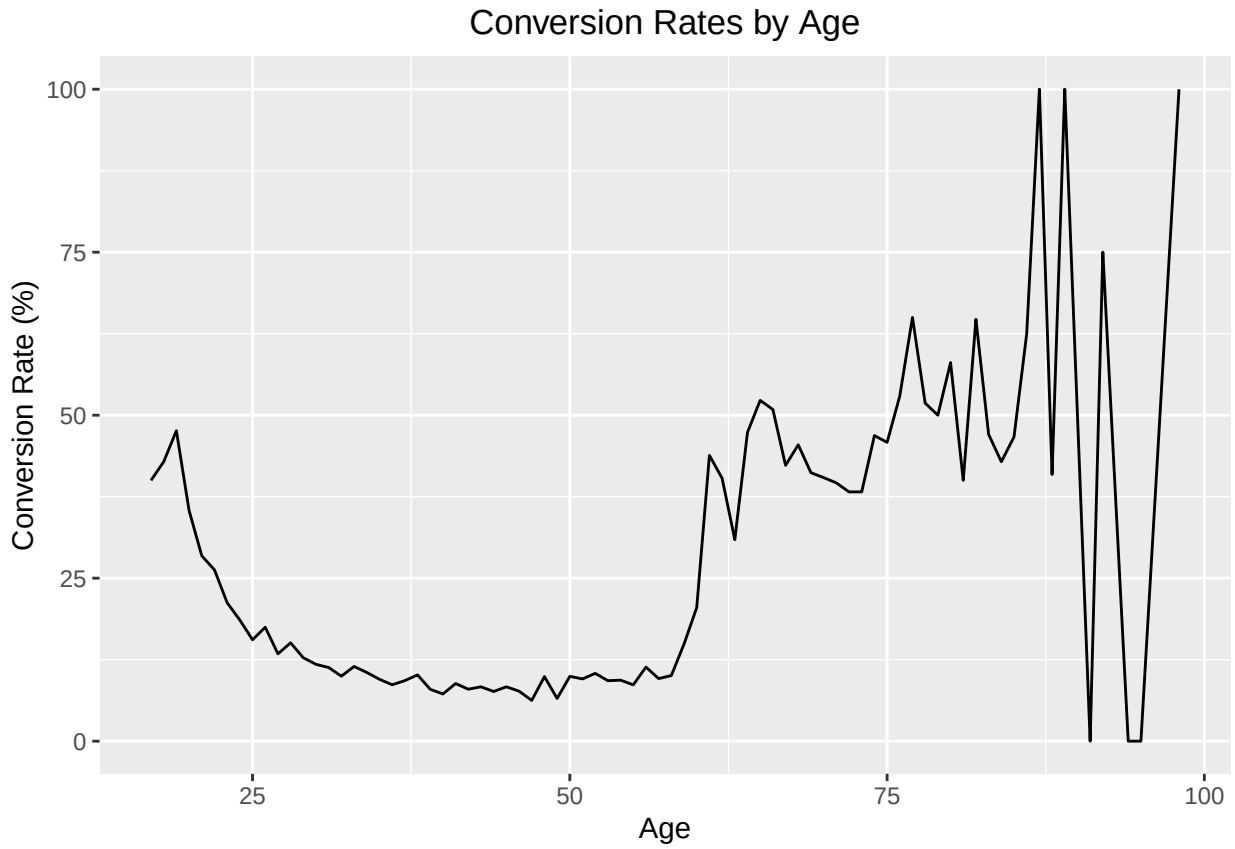
```
# a. by age
conversionsByAge <- conversionsDF %>%
  group_by(Age=age) %>%
  summarise(TotalCount=n(), NumConversions=sum(conversion)) %>%
  mutate(ConversionRate=NumConversions/TotalCount*100.0)

head(conversionsByAge)
```

```
## # A tibble: 6 x 4
##     Age TotalCount NumConversions ConversionRate
##   <int>      <int>          <int>          <dbl>
## 1    17          5              2           40
## 2    18         28             12           42.9
## 3    19         42             20           47.6
## 4    20         65             23           35.4
## 5    21        102             29           28.4
## 6    22        137             36           26.3
```

Another way to look at conversion rates across client ages is by plotting a line chart, as shown in the following screenshot:

Note: We are using the ggplot function to initialize a ggplot object with conversionsByAge as the data and the column, Age, as the x-axis and the column, ConversionRate, as the y-axis. Then, we use geom_line function to connect the observations and create a line chart. You can change the title of a plot, by using ggtitle function. Also, you can use xlab and ylab functions to rename the x-axis label and y-axis label respectively.

```
# line chart
ggplot(data=conversionsByAge, aes(x=Age, y=ConversionRate)) +
  geom_line() +
  ggtitle('Conversion Rates by Age') +
  xlab("Age") +
  ylab("Conversion Rate (%)") +
  theme(plot.title = element_text(hjust = 0.5))
```



One thing that is noticeable in the previous line chart is the fact that there seems to be lots of noise in older age groups. Conversion rates for those who are 70 or older vary a lot and if we look at the data, this is mostly because the number of clients in this age group is relatively small, compared to other age groups.

In order to reduce this unwanted noise, we can group multiple ages together. In the section below, we group bank clients into six different groups, based on their age—between 18 and 30, between 30 and 40, between 40 and 50, between 50 and 60, between 60 and 70, and 70 and older. The following code can be used to group the clients into their corresponding groups:

```r
# b. by age groups
conversionsByAgeGroup <- conversionsDF %>%
  group_by(AgeGroup=cut(age, breaks= seq(20, 70, by = 10)) ) %>%
  summarise(TotalCount=n(), NumConversions=sum(conversion)) %>%
  mutate(ConversionRate=NumConversions/TotalCount*100.0)

conversionsByAgeGroup$AgeGroup <- as.character(conversionsByAgeGroup$AgeGroup)
conversionsByAgeGroup$AgeGroup[6] <- "70+"
```

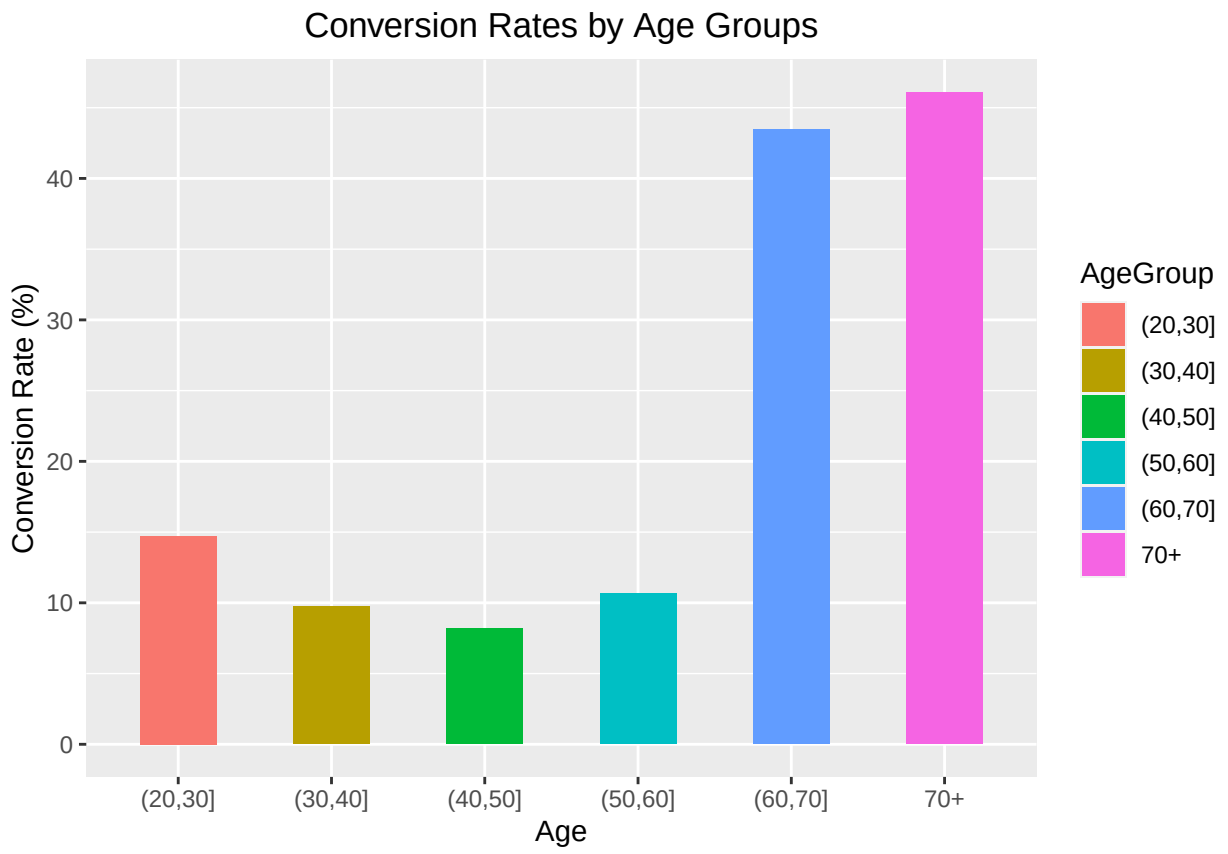
```r
conversionsByAgeGroup
```

```
## # A tibble: 6 x 4
##   AgeGroup TotalCount NumConversions ConversionRate
##   <chr>         <int>          <int>          <dbl>
## 1 (20,30]        7243           1067           14.7
## 2 (30,40]       16385           1597            9.75
## 3 (40,50]       10240            837            8.17
## 4 (50,60]        6270            668           10.7
## 5 (60,70]         488            212           43.4
## 6 70+             562            259           46.1
```

As with the previous case, we are using the group_by function to group the conversionsDF data by the age column. The difference here is how we used the cut function to create the age range for each age group.

The breaks argument defines the points at which the cut function is going to divide the DataFrame. The argument, seq(20, 70, by = 10), means we are going to create a sequence from 20 to 70 in increments of 10. Once the data is grouped by these age groups, the rest are the same as before. We are using the summarise and mutate functions to compute for the TotalCount, NumConversions, and ConversionRate columns.

```r
# bar chart
ggplot(conversionsByAgeGroup, aes(x=AgeGroup, y=ConversionRate, fill=AgeGroup)) +
  geom_bar(width=0.5, stat="identity") +
  ggtitle('Conversion Rates by Age Groups') +
  xlab("Age") +
  ylab("Conversion Rate (%)") +
  theme(plot.title = element_text(hjust = 0.5))
```

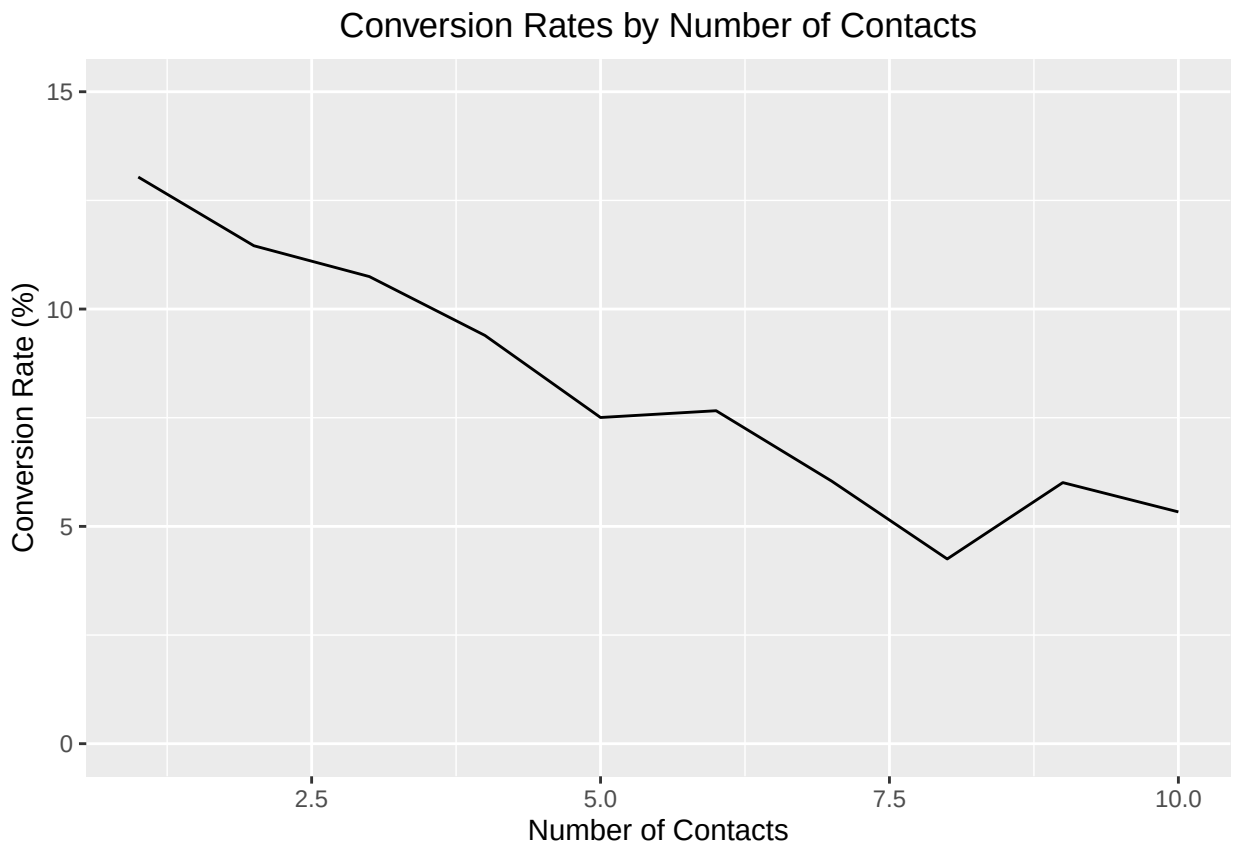


3. Conversion Rates by Number of Contacts

```
conversionsByNumContact <- conversionsDF %>%
  group_by(NumContact=campaign) %>%
  summarise(TotalCount=n(), NumConversions=sum(conversion)) %>%
  mutate(ConversionRate=NumConversions/TotalCount*100.0)

head(conversionsByNumContact, 10)
```

```
## # A tibble: 10 x 4
##    NumContact TotalCount NumConversions ConversionRate
##         <int>      <int>          <int>          <dbl>
##  1          1      17642           2300          13.0
##  2          2      10570           1211          11.5
##  3          3       5341            574          10.7
##  4          4       2651            249           9.39
##  5          5       1599            120           7.50
##  6          6        979             75           7.66
##  7          7        629             38           6.04
##  8          8        400             17           4.25
##  9          9        283             17           6.01
## 10         10        225             12           5.33
```

```r
# line chart
ggplot(data=head(conversionsByNumContact, 10), aes(x=NumContact, y=ConversionRate)) +
  geom_line() +
  ggtitle('Conversion Rates by Number of Contacts') +
  xlab("Number of Contacts") +
  ylab("Conversion Rate (%)") +
  ylim(c(0, 15)) +
  theme(plot.title = element_text(hjust = 0.5))
```



4. Conversions vs. Non-Conversions

One other thing we can look at is the demographic differences between the converted clients and non-converted clients. This type of analysis can help us identify what differentiates converted groups from non-converted groups in our marketing campaigns and helps us understand our target clients better and what types of customers respond better to our marketing efforts. In this exercise, we will compare the distributions of the marital status among the conversions and non-conversions groups.

We will first count the number of conversions and non-conversions for each marital status. The following code shows how we can compute this using R functions:

```r
# 4.1. Marital Status
conversionsByMaritalStatus <- conversionsDF %>%
  group_by(Marital=marital, Conversion=conversion) %>%
  summarise(Count=n())
```

```
## `summarise()` has grouped output by 'Marital'. You can override using the `.groups` argument.
```

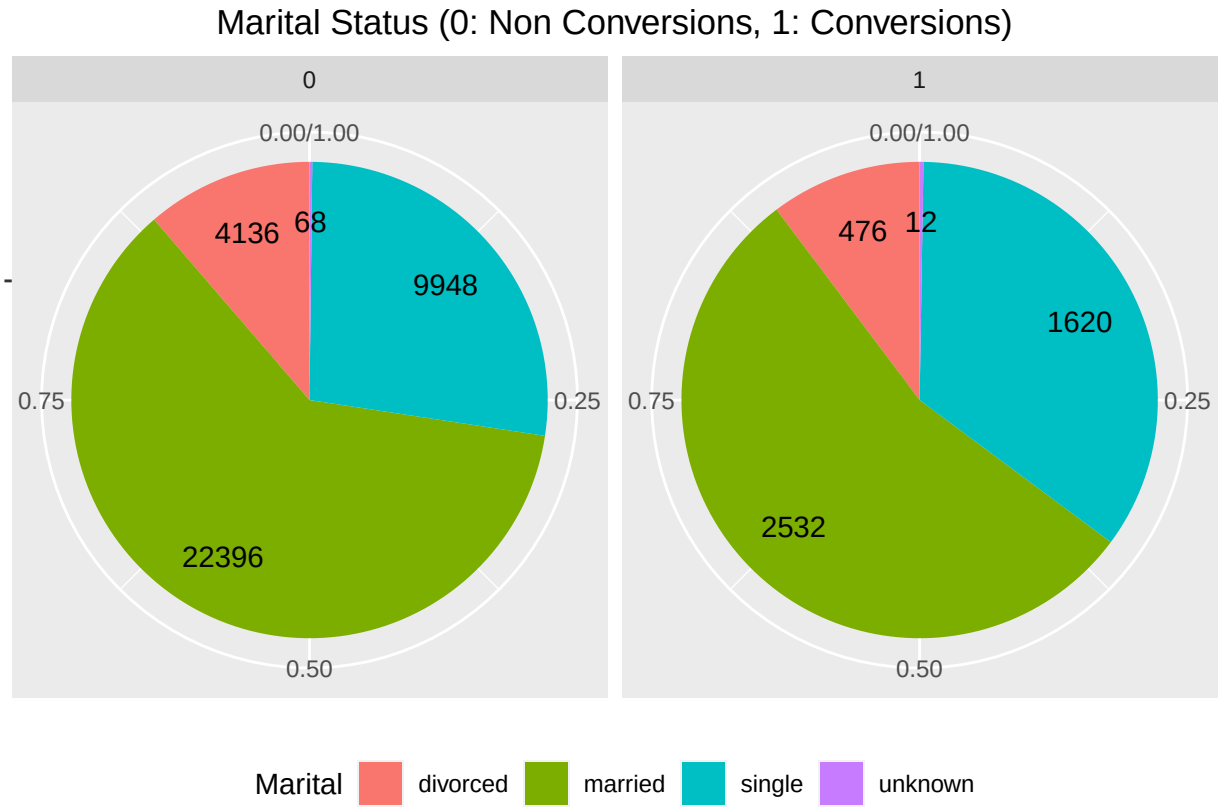
```
conversionsByMaritalStatus
```

```
## # A tibble: 8 x 3
## # Groups:   Marital [4]
##   Marital  Conversion Count
##   <chr>         <int> <int>
## 1 divorced          0  4136
## 2 divorced          1   476
## 3 married           0 22396
## 4 married           1  2532
## 5 single            0  9948
## 6 single            1  1620
## 7 unknown           0    68
## 8 unknown           1    12
```

Note: As we can see from the above code, we are using the pipe operator, %>%, in the dplyr package to pass the DataFrame, conversionsDF to the group_by function and then to the summarise function. In the group_by function, we are grouping by two columns, marital and conversion. In the summarise function, we are simply counting the number of records in each group, by using the n function.

Another way to present the above data by pie-chart:

```
# pie chart
ggplot(conversionsByMaritalStatus, aes(x="", y=Count, fill=Marital)) +
  geom_bar(width=1, stat = "identity", position=position_fill()) +
  geom_text(aes(x=1.25, label=Count), position=position_fill(vjust = 0.5)) +
  coord_polar("y") +
  facet_wrap(~Conversion) +
  ggtitle('Marital Status (0: Non Conversions, 1: Conversions)') +
  theme(
    axis.title.x=element_blank(),
    axis.title.y=element_blank(),
    plot.title=element_text(hjust=0.5),
    legend.position='bottom'
  )
```



Note: For building a pie chart in R, we are using the same geom_bar function, just as if we are building a bar chart. The difference here is coord_polar("y"), which transforms a bar chart into a pie chart. Then, we are using the facet_wrap function to create two columns of pie charts by the column, Conversion. This builds two pie charts, one for the conversions group and another for the non-conversions group.

Compared to the tabular format of the data output, pie charts make it much easier to understand the overall distributions of the data. With pie charts, we can easily see that the married group takes up the largest proportions in both conversions and non-conversions groups, while the single group comes second. Using pie charts, we can easily visualize the similarities and differences between two groups.

```
# 4.2. Education
conversionsByEducation <- conversionsDF %>%
  group_by(Education=education, Conversion=conversion) %>%
  summarise(Count=n())
```

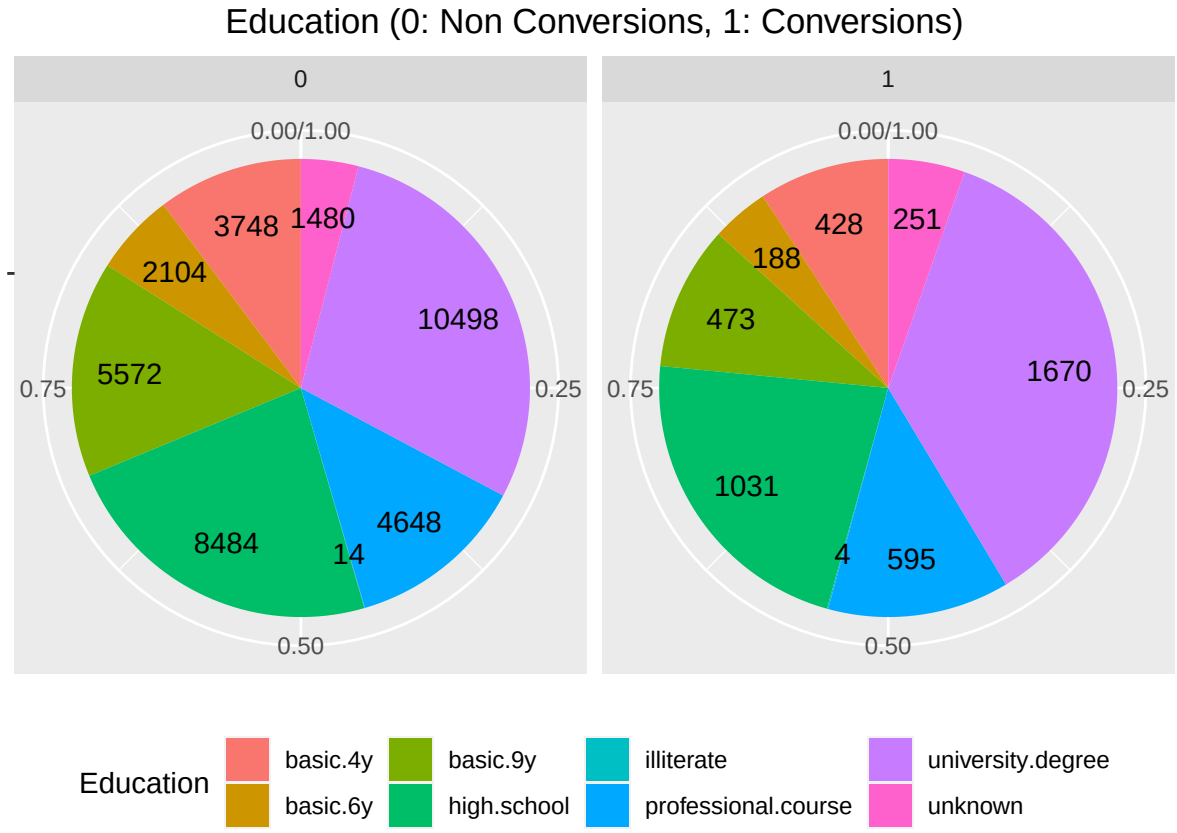
```
## `summarise()` has grouped output by 'Education'. You can override using the `.groups` argument.
```

```
conversionsByEducation
```

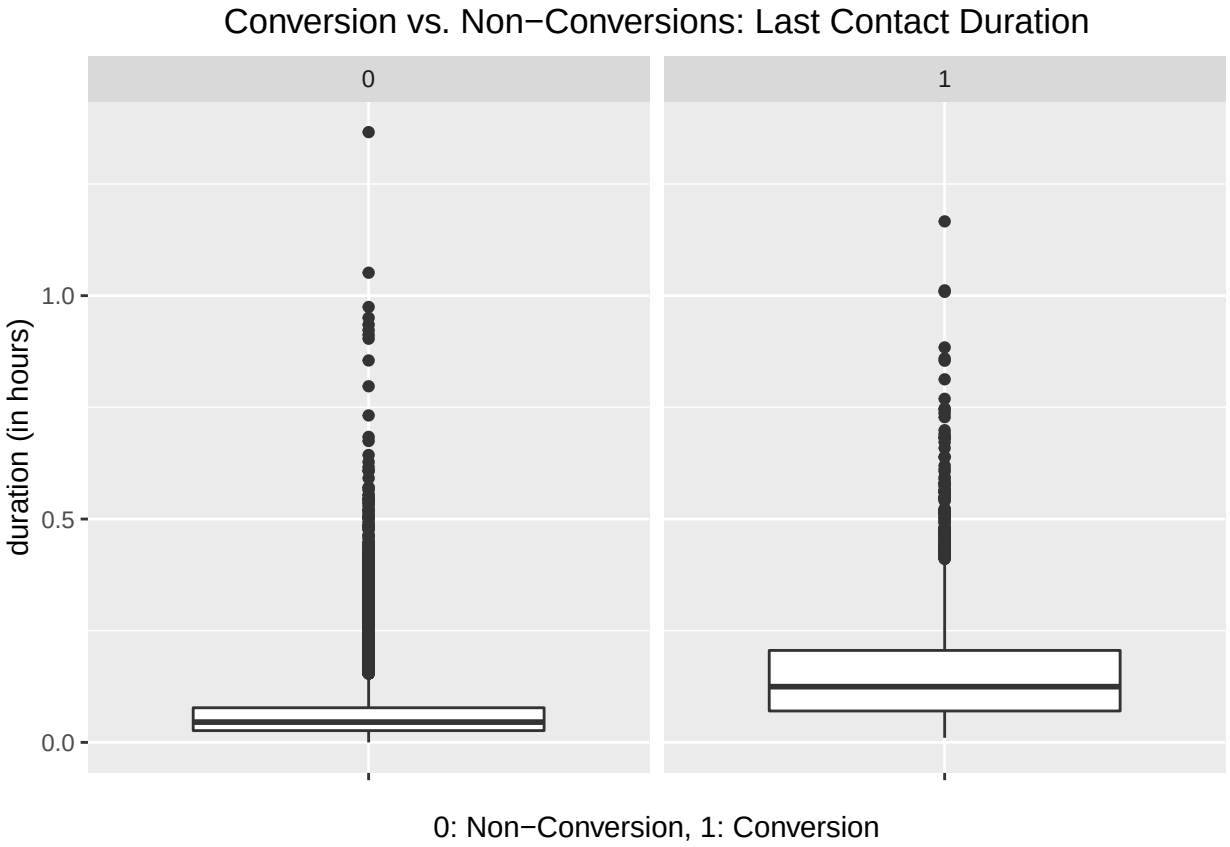
```
## # A tibble: 16 x 3
## # Groups:   Education [8]
##    Education           Conversion Count
##    <chr>                    <int> <int>
##  1 basic.4y                     0  3748
##  2 basic.4y                     1   428
##  3 basic.6y                     0  2104
```

```
##  4 basic.6y                    1   188
##  5 basic.9y                    0  5572
##  6 basic.9y                    1   473
##  7 high.school                 0  8484
##  8 high.school                 1  1031
##  9 illiterate                  0    14
## 10 illiterate                  1     4
## 11 professional.course         0  4648
## 12 professional.course         1   595
## 13 university.degree           0 10498
## 14 university.degree           1  1670
## 15 unknown                     0  1480
## 16 unknown                     1   251
```

```r
# pie chart
ggplot(conversionsByEducation, aes(x="", y=Count, fill=Education)) +
  geom_bar(width=1, stat = "identity", position=position_fill()) +
  geom_text(aes(x=1.25, label=Count), position=position_fill(vjust = 0.5)) +
  coord_polar("y") +
  facet_wrap(~Conversion) +
  ggtitle('Education (0: Non Conversions, 1: Conversions)') +
  theme(
    axis.title.x=element_blank(),
    axis.title.y=element_blank(),
    plot.title=element_text(hjust=0.5),
    legend.position='bottom'
  )
```




```
# 4.3. Last Contact Duration
conversionsDF$duration <- conversionsDF$duration / (60*60)

ggplot(conversionsDF, aes(x="", y=duration)) +
  geom_boxplot() +
  facet_wrap(~conversion) +
  ylab("duration (in hours)") +
  xlab("0: Non-Conversion, 1: Conversion") +
  ggtitle("Conversion vs. Non-Conversions: Last Contact Duration") +
  theme(plot.title=element_text(hjust=0.5))
```



5. Conversions by Age Groups & Marital Status

So far, we have aggregated our data by one criterion. However, there are cases where you want to group the data by more than one column. In this section, we will discuss how we can analyze and report conversion rates by more than one criterion. In this section, we will use the age groups that we have built in the previous section and the marital status as the two columns to group by.

Let's first look at the code:

```
conversionsByAgeMarital <- conversionsDF %>%
  group_by(AgeGroup=cut(age, breaks= seq(20, 70, by = 10)), Marital=marital) %>%
  summarise(Count=n(), NumConversions=sum(conversion)) %>%
  mutate(TotalCount=sum(Count)) %>%
  mutate(ConversionRate=NumConversions/TotalCount)
```

```
## `summarise()` has grouped output by 'AgeGroup'. You can override using the `.groups` argument.
```

```
conversionsByAgeMarital$AgeGroup <- as.character(conversionsByAgeMarital$AgeGroup)
conversionsByAgeMarital$AgeGroup[is.na(conversionsByAgeMarital$AgeGroup)] <- "70+"
conversionsByAgeMarital
```

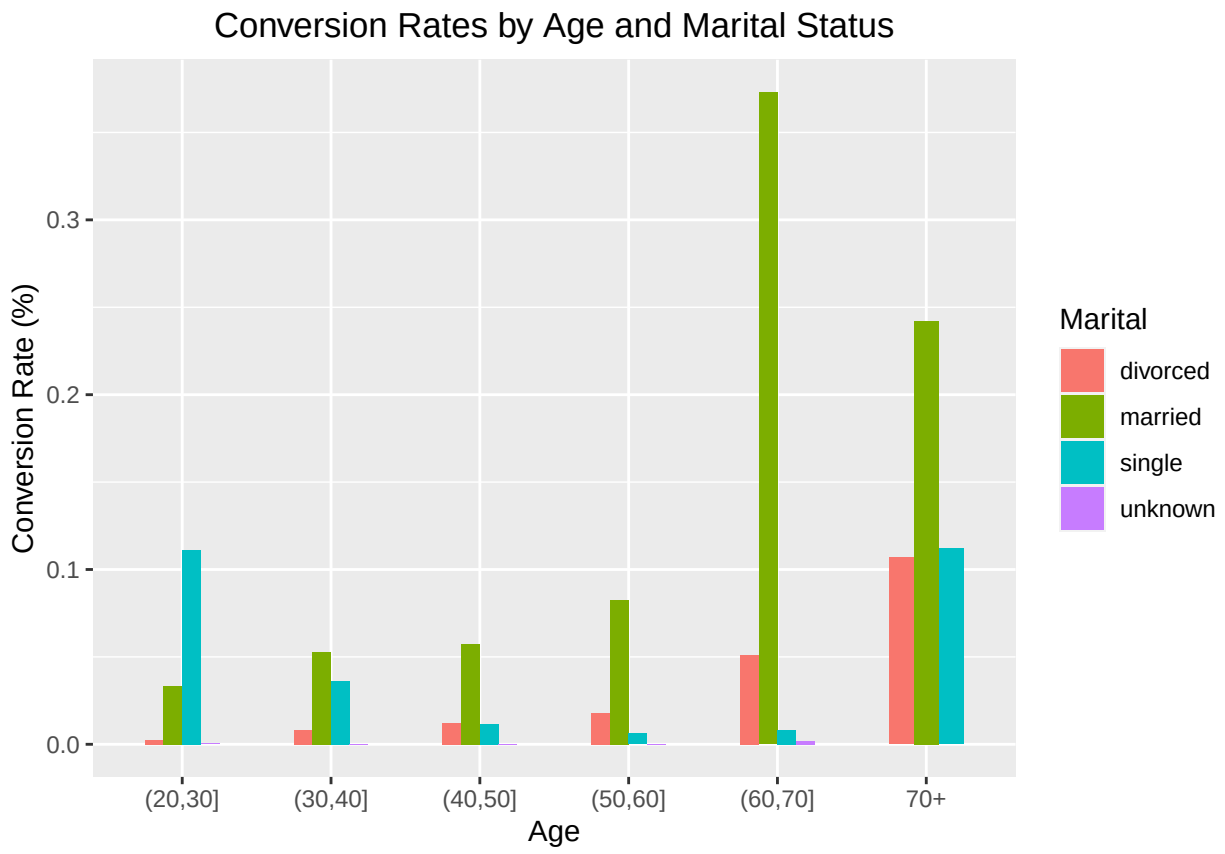
```
## # A tibble: 23 x 6
## # Groups:   AgeGroup [6]
##    AgeGroup Marital  Count NumConversions TotalCount ConversionRate
```

```
##    <chr>    <chr>    <int>          <int>      <int>          <dbl>
##  1 (20,30]  divorced   229             18       7243       0.00249
##  2 (20,30]  married   2389            242       7243       0.0334
##  3 (20,30]  single    4612            804       7243       0.111
##  4 (20,30]  unknown     13              3       7243       0.000414
##  5 (30,40]  divorced  1505            135      16385       0.00824
##  6 (30,40]  married   9705            867      16385       0.0529
##  7 (30,40]  single    5139            591      16385       0.0361
##  8 (30,40]  unknown     36              4      16385       0.000244
##  9 (40,50]  divorced  1548            126      10240       0.0123
## 10 (40,50]  married   7383            588      10240       0.0574
## # ... with 13 more rows
```

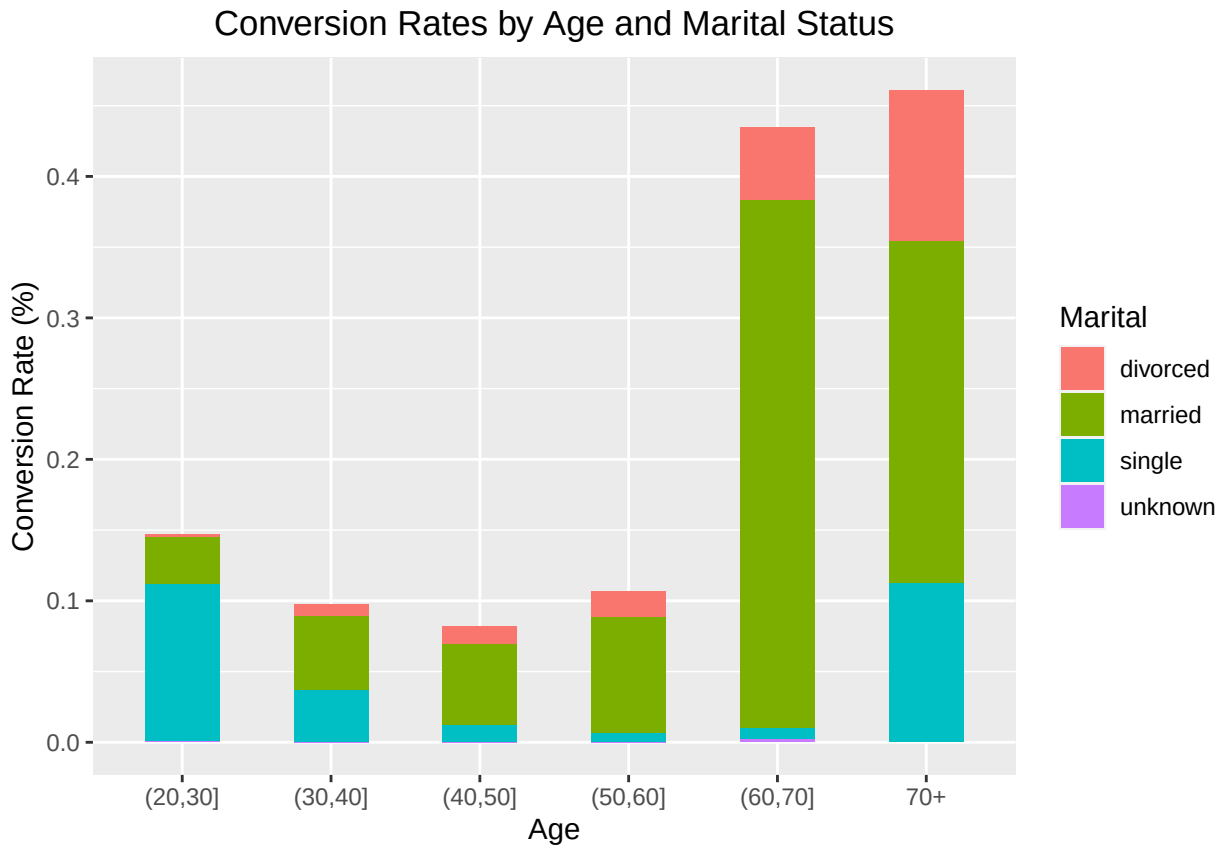
Note: Similar to when we built custom age groups, we are using the cut function in group_by to create age groups from 20 to 70 in increments of 10. However, "we are grouping by the column,marital, as well this time.

Then, we are using the summarise function to compute the number of records in each group Count, and the number of conversions in each group, NumConversions. Then, using the mutate function, we calculate the total counts in each age group, named TotalCount, and the conversion rates in each group, named ConversionRate.

```
# bar chart
ggplot(conversionsByAgeMarital, aes(x=AgeGroup, y=ConversionRate, fill=Marital)) +
  geom_bar(width=0.5, stat="identity", position="dodge") +
  ylab("Conversion Rate (%)") +
  xlab("Age") +
  ggtitle("Conversion Rates by Age and Marital Status") +
  theme(plot.title=element_text(hjust=0.5))
```




```
# stacked bar chart
ggplot(conversionsByAgeMarital, aes(x=AgeGroup, y=ConversionRate, fill=Marital)) +
  geom_bar(width=0.5, stat="identity", position="stack") +
  ylab("Conversion Rate (%)") +
  xlab("Age") +
  ggtitle("Conversion Rates by Age and Marital Status") +
  theme(plot.title=element_text(hjust=0.5))
```



Summary: These above are various KPIs that are often used in marketing to track the progress of marketing campaigns. We have learned how important it is to look at how much sales revenue each marketing strategy generates. When analyzing the sales revenue metrics, we have seen that it is important to approach it from different angles. We might want to look at not only the aggregate sales revenue, but also time-series (monthly, quarterly, or yearly) sales revenue. We might also want to look at sales attributed to each individual marketing campaigns and how much revenue each campaign generated for the company. We will go other sections with various metrics to analyze for digital marketing channels as well, such as CPA, CTR, lead ratio, and conversion rates.