Term Deposit Prediction

Julien Roquelaure

2021/01/07

Introduction

The data set is extracted from the UCI Machine Learning Repository.

The data is about marketing campaigns made by a Portuguese banking institution. The classification goal is to predict if phone calls will result in a client subscribing a term deposit.

The data is comprised of 41188 instances of 20 input variables and 1 output variable.

The input data is as follows:

7 variables about the client:

- *age*: integer
- *job*: type of job among 12 categories
- *marital*: 4 categories of marital status
- *education*: 8 categories
- *default*: client has credit in default? yes/no/unknown
- *housing*: client has housing loan? yes/no/unknown
- *loan*: client has personal loan? yes/no/unknown

4 variables about the last contact of the current campaign:

- *contact*: 2 categories of phone
- *month*: 12 months
- *day_of_week*: 5 working days
- *duration*: duration of the call

4 variables about campaigns:

- *campaign*: number of calls made to the client during the current campaign
- *pdays*: interval in days between previous campaign and last campaign, 999 meaning no previous contact
- *previous*: number of calls during the previous campaign
- *poutcome*: 3 categories of outcome for previous campaign

5 social and economic variables:

- *emp.var.rate*: employment variation rate (quarterly)
- *cons.price.idx*: consumer price index (monthly)
- *cons.conf.idx*: consumer confidence index (monthly)
- *euribor3m*: euribor 3 months rate (daily)
- *nr.employed*: number of employees of the company (quarterly)

There is also 1 output variable:

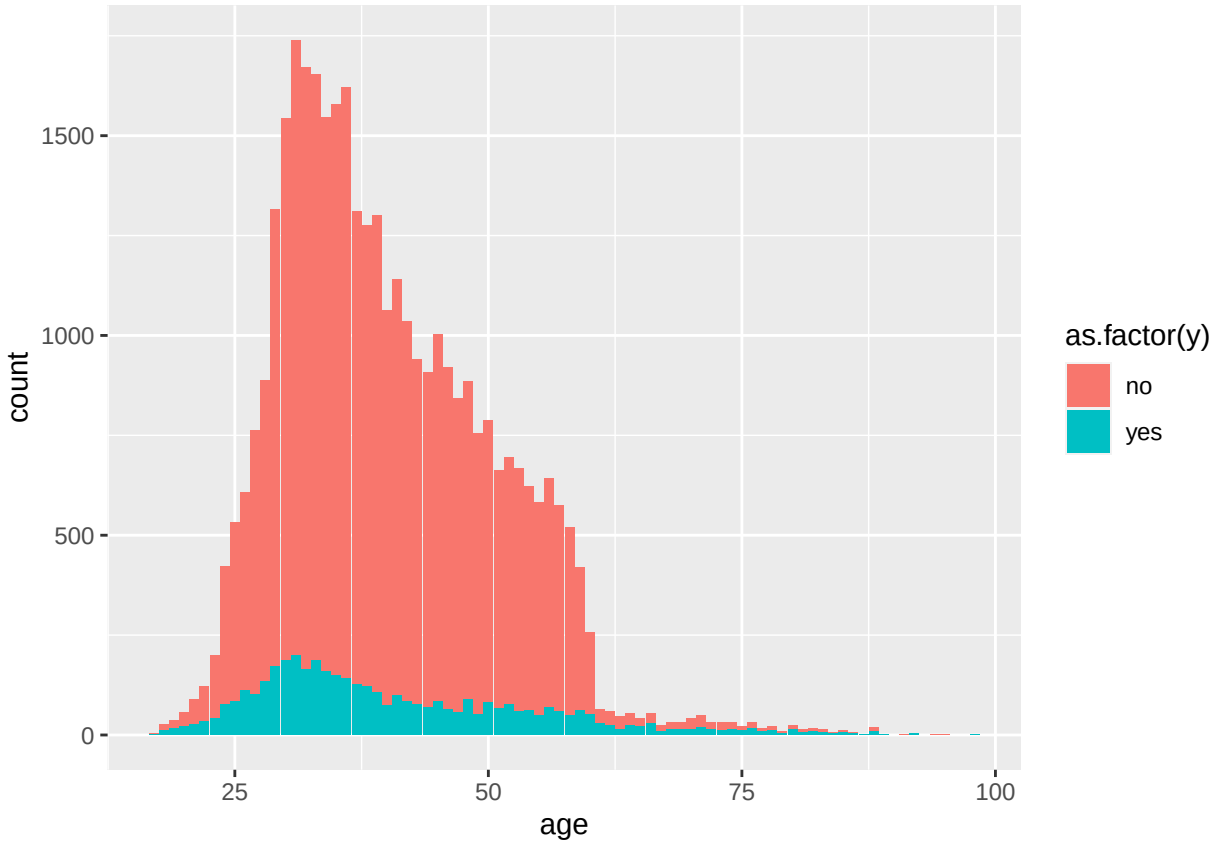
- y: client has subscribed a term deposit? yes/no

Analysis

We have 20 variables. So our first decision is to analyze them one by one. And after that, we are going to perform a linear regression and see if we were right about our decision of discarding the variables.

Age

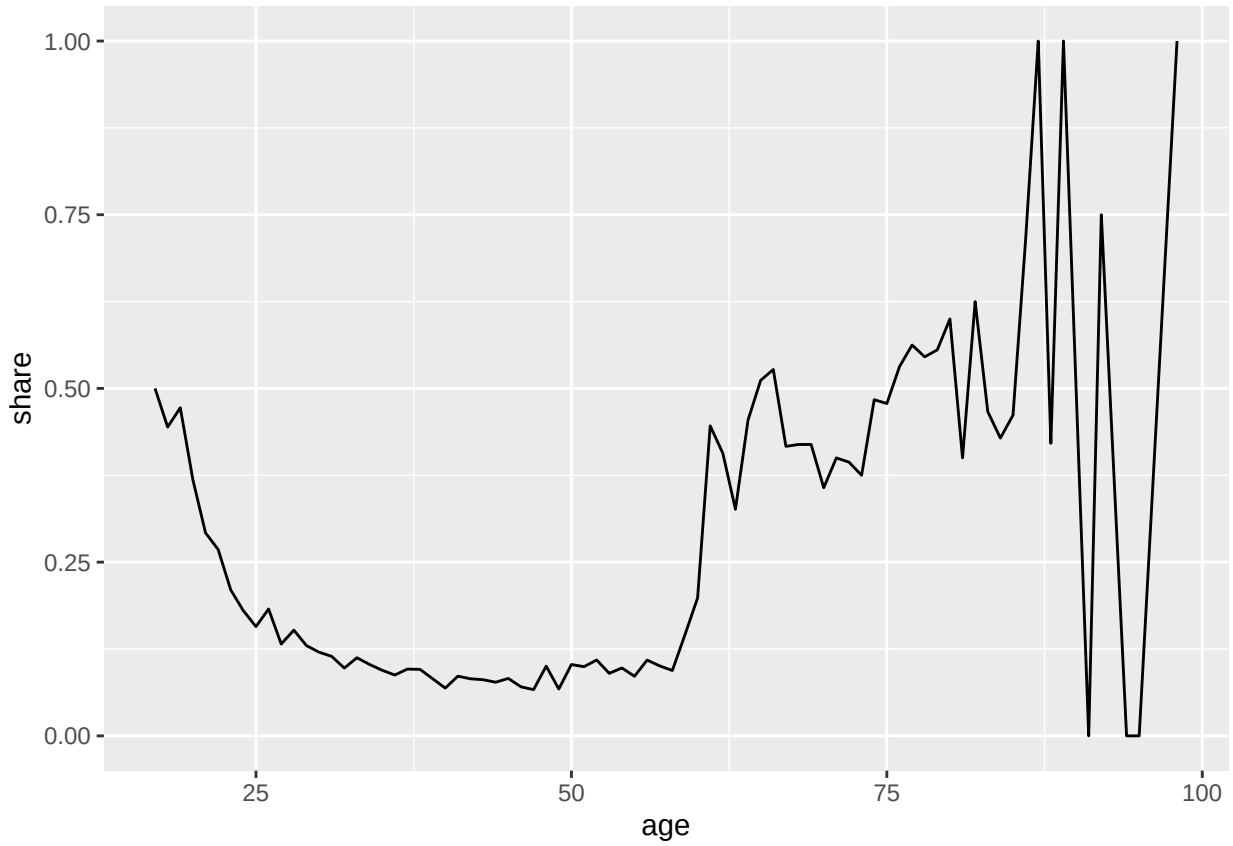
```
calls_train %>%
  ggplot(aes(age, fill=as.factor(y))) +
  geom_bar()
```



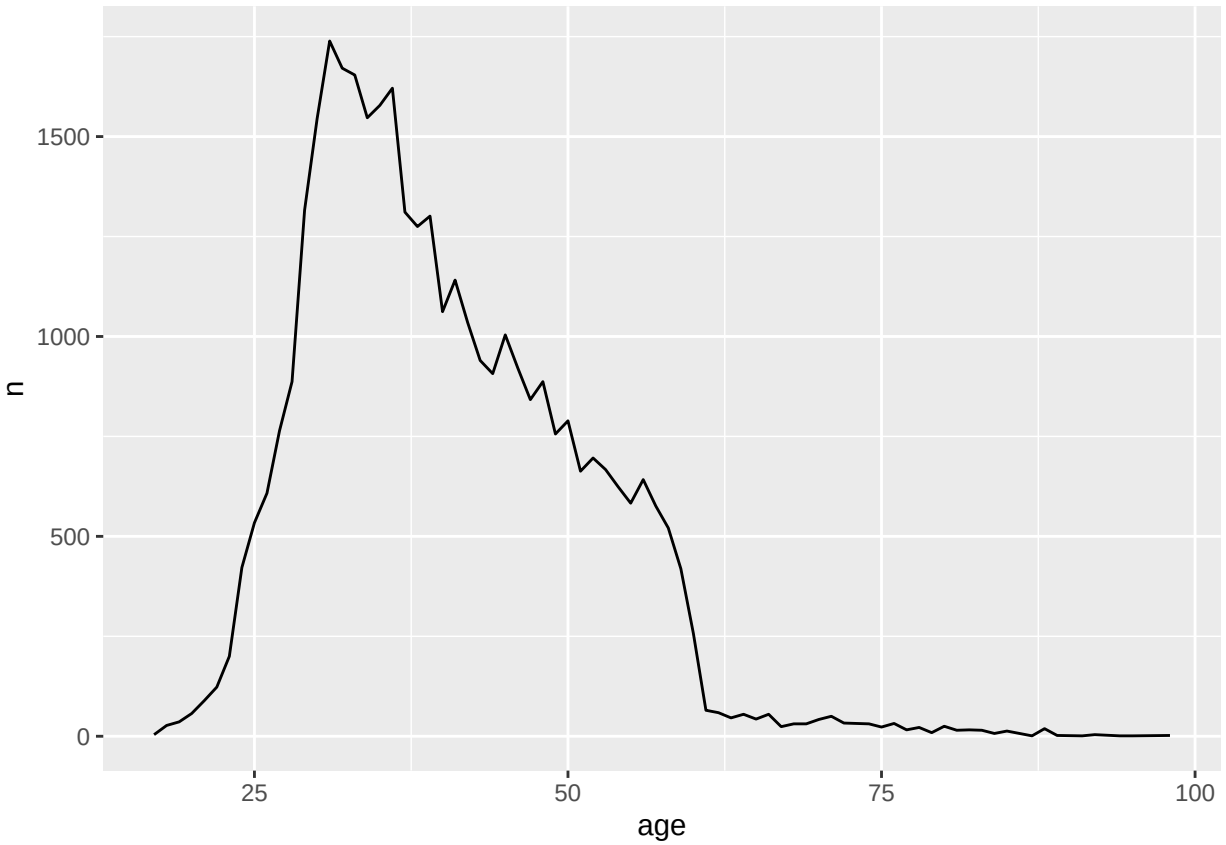

```
age_structure <-calls_train %>%
  group_by(age) %>%
  summarize(n=n(), share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
age_structure %>%
  ggplot(aes(x=age, y=share)) + geom_line()
```



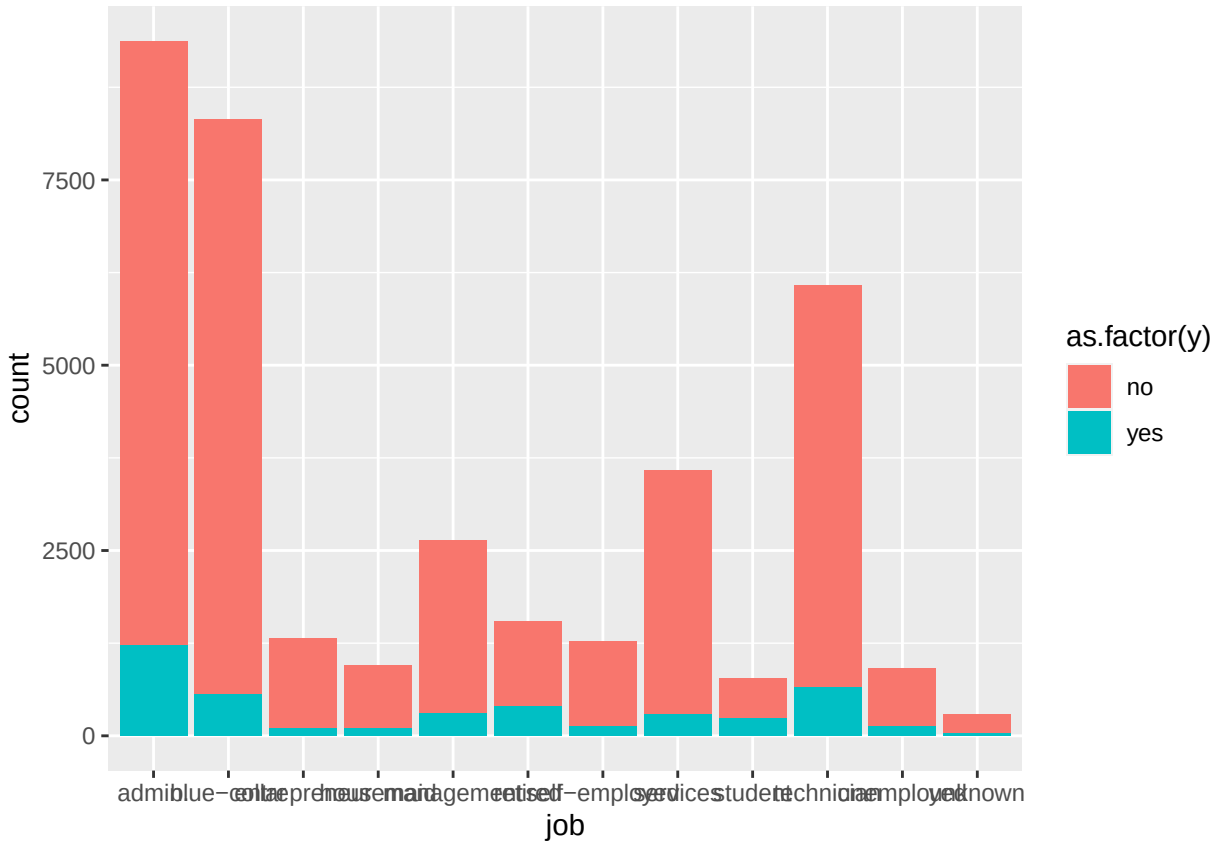

```
age_structure %>%
  ggplot(aes(x=age, y=n)) + geom_line()
```



The high and low values predict high subscription rate. But their counts are very low. So we will normalize them with a square distance to the mean, but we will expect to discard it in the end.

Job

```
calls_train %>%
  ggplot(aes(job, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(job) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

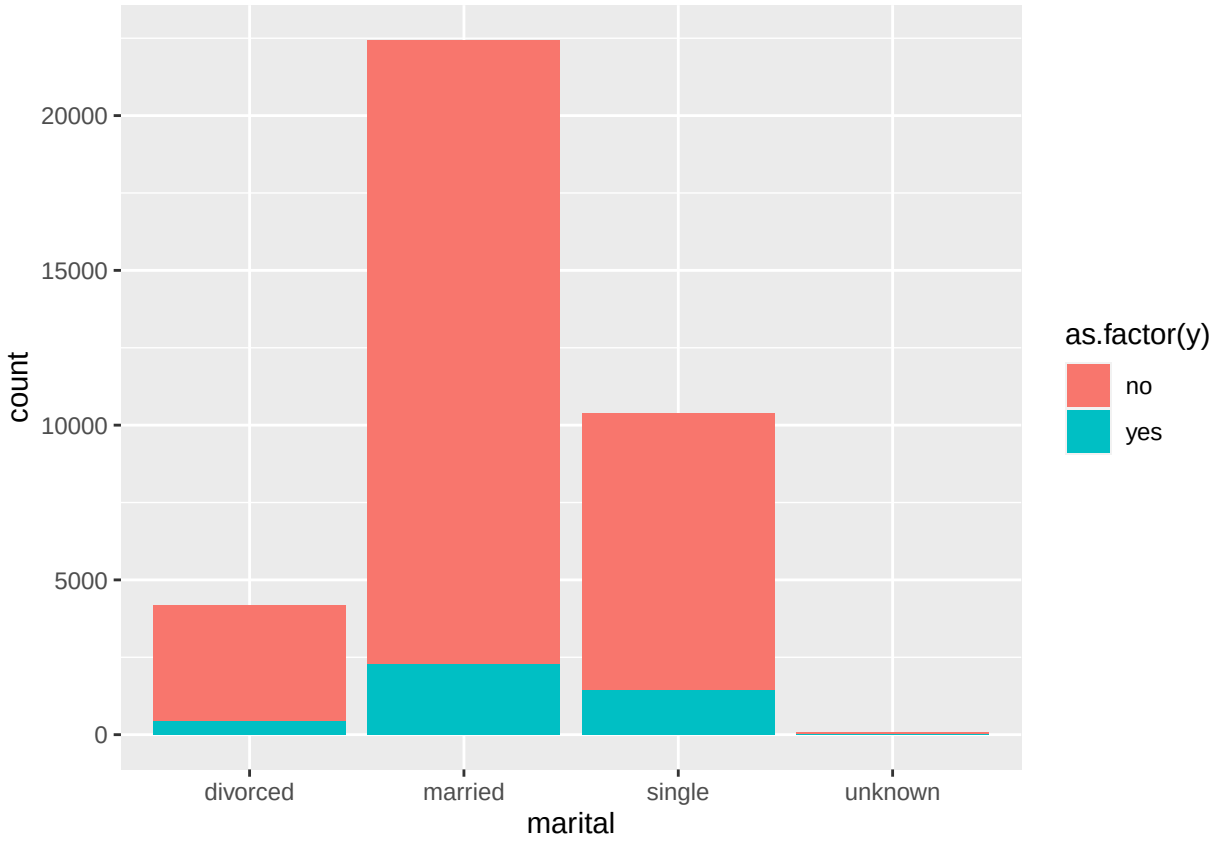
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 12 x 2
##    job            share
##    <chr>          <dbl>
##  1 admin.        0.130
##  2 blue-collar   0.0679
##  3 entrepreneur  0.0824
##  4 housemaid     0.103
##  5 management    0.115
##  6 retired       0.255
##  7 self-employed 0.103
##  8 services      0.0809
##  9 student       0.313
## 10 technician    0.109
## 11 unemployed    0.146
## 12 unknown       0.103
```

We see high rates for students and retired, which is expected from the age variable. There is no clear pattern among the categories and we found no satisfying way of converting it into a numerical variable so we will just use admin.? yes/no as a binary variable but we don't expect much.

Marital

```
calls_train %>%
  ggplot(aes(marital, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(marital) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

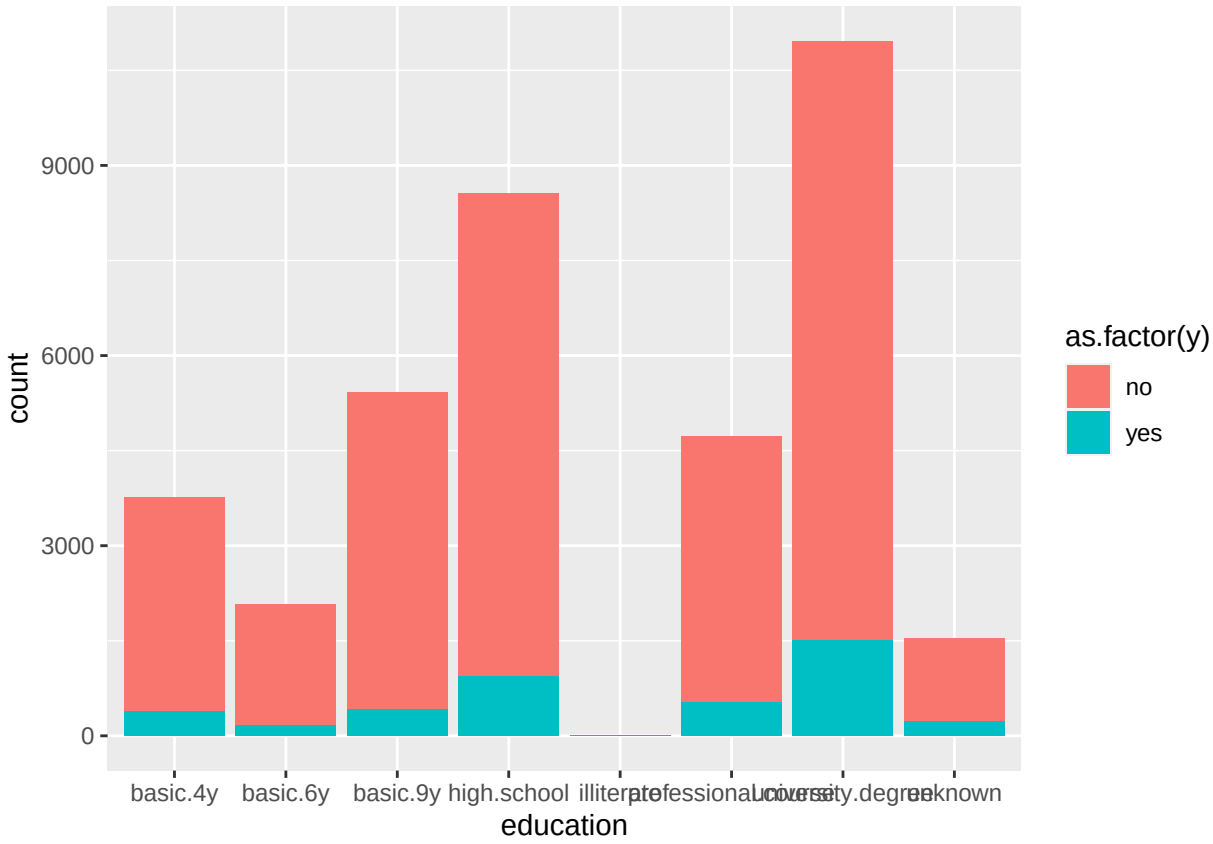
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 4 x 2
##   marital  share
##   <chr>    <dbl>
## 1 divorced 0.101
## 2 married  0.102
## 3 single   0.140
## 4 unknown  0.167
```

Given the small number of unknowns, we will consider a binary variable single? yes/no.

Education

```
calls_train %>%
  ggplot(aes(education, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(education) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

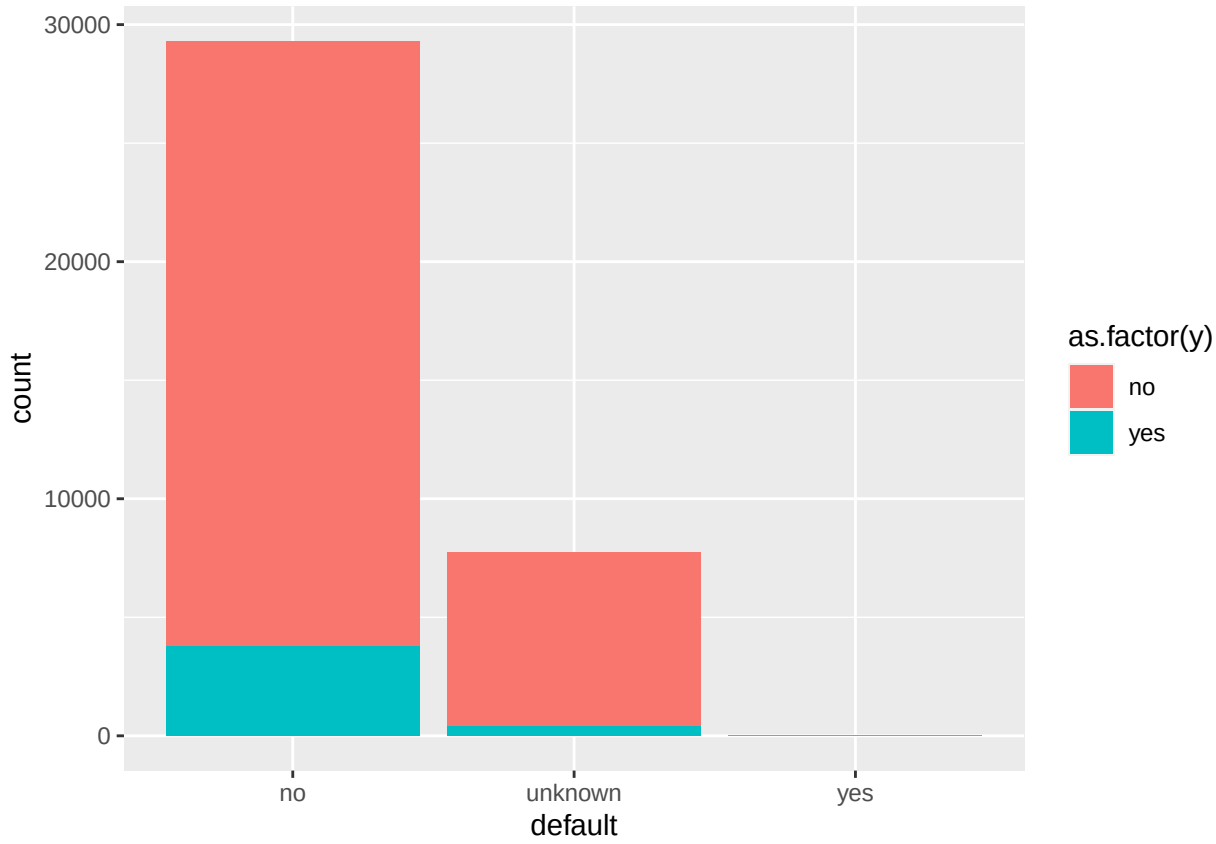
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   education            share
##   <chr>                <dbl>
## 1 basic.4y             0.102
## 2 basic.6y             0.0812
## 3 basic.9y             0.0763
## 4 high.school          0.109
## 5 illiterate           0.25
## 6 professional.course  0.113
## 7 university.degree    0.138
## 8 unknown              0.147
```

The number of illiterates is very small and there seems to be no pattern. We will binarize university.degree? yes/no with no expectation.

Default

```
calls_train %>%
  ggplot(aes(default, fill=as.factor(y))) +
  geom_bar()
```




```
sum(calls$default == "yes")
```

```
## [1] 3
```

```
calls_train %>%
  group_by(default) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

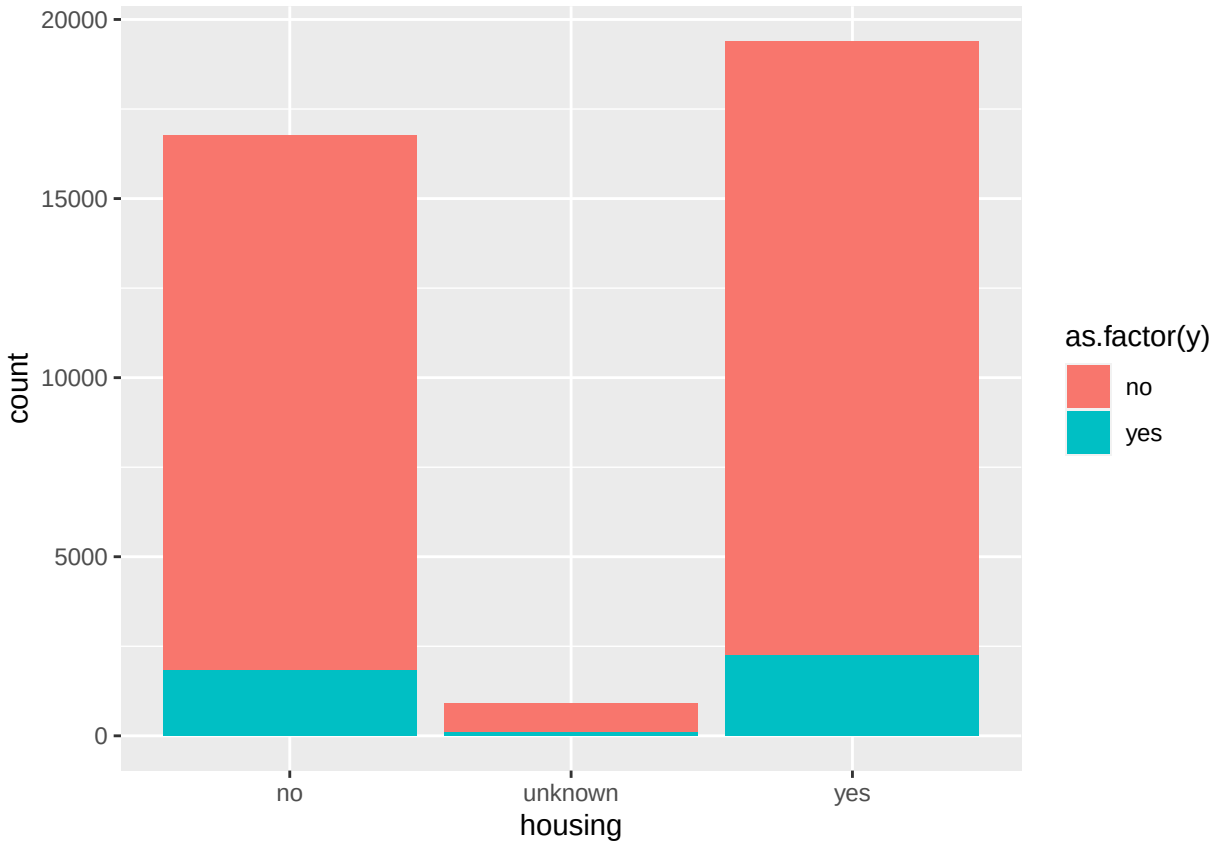
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2
##   default  share
##   <chr>    <dbl>
## 1 no      0.129
## 2 unknown 0.0509
## 3 yes     0
```

The number of "yes" is only 3. We decide to binarize the variable with "no" = 0 and "unknown"/"yes" = 1

Housing

```
calls_train %>%
  ggplot(aes(housing, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(housing) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

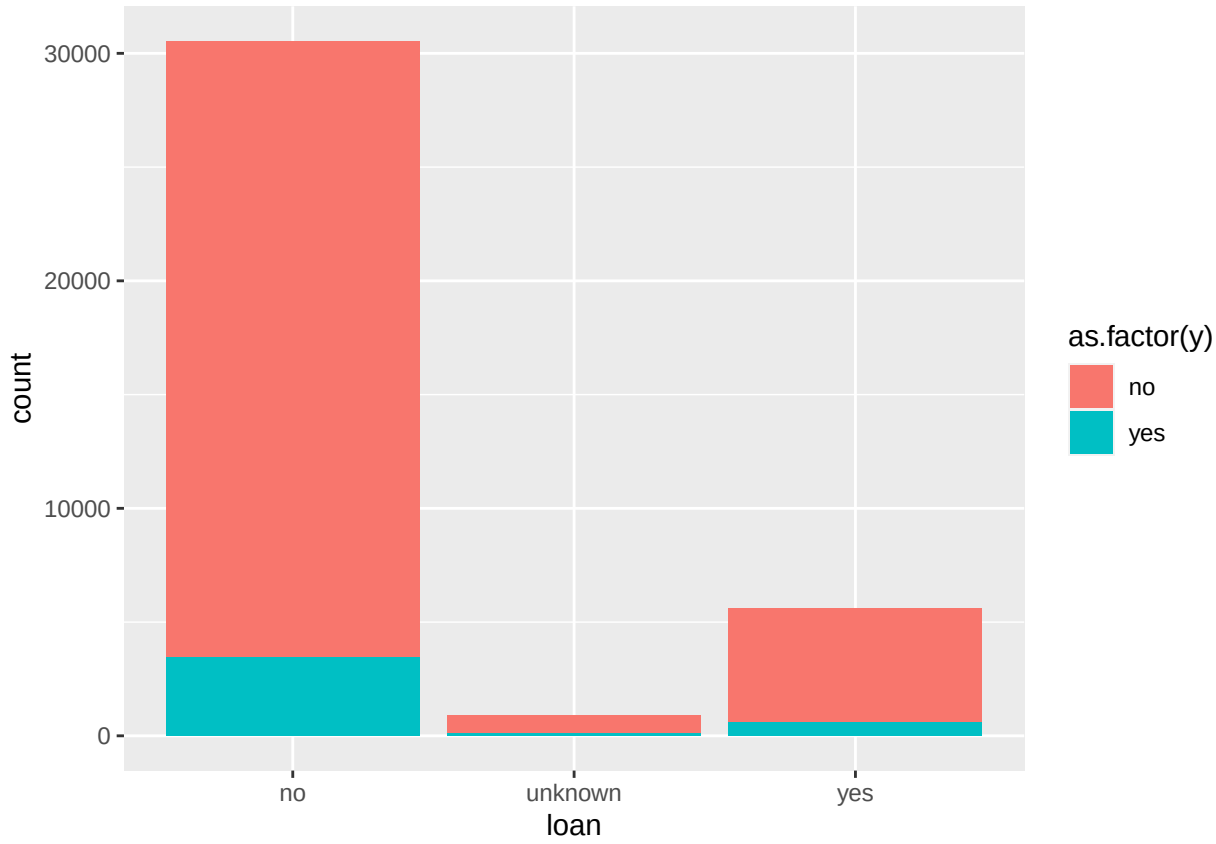
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2
##   housing share
##   <chr>   <dbl>
## 1 no      0.108
## 2 unknown 0.110
## 3 yes     0.116
```

Nothing remarkable, we binarize.

Loan

```
calls_train %>%
  ggplot(aes(loan, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(loan) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

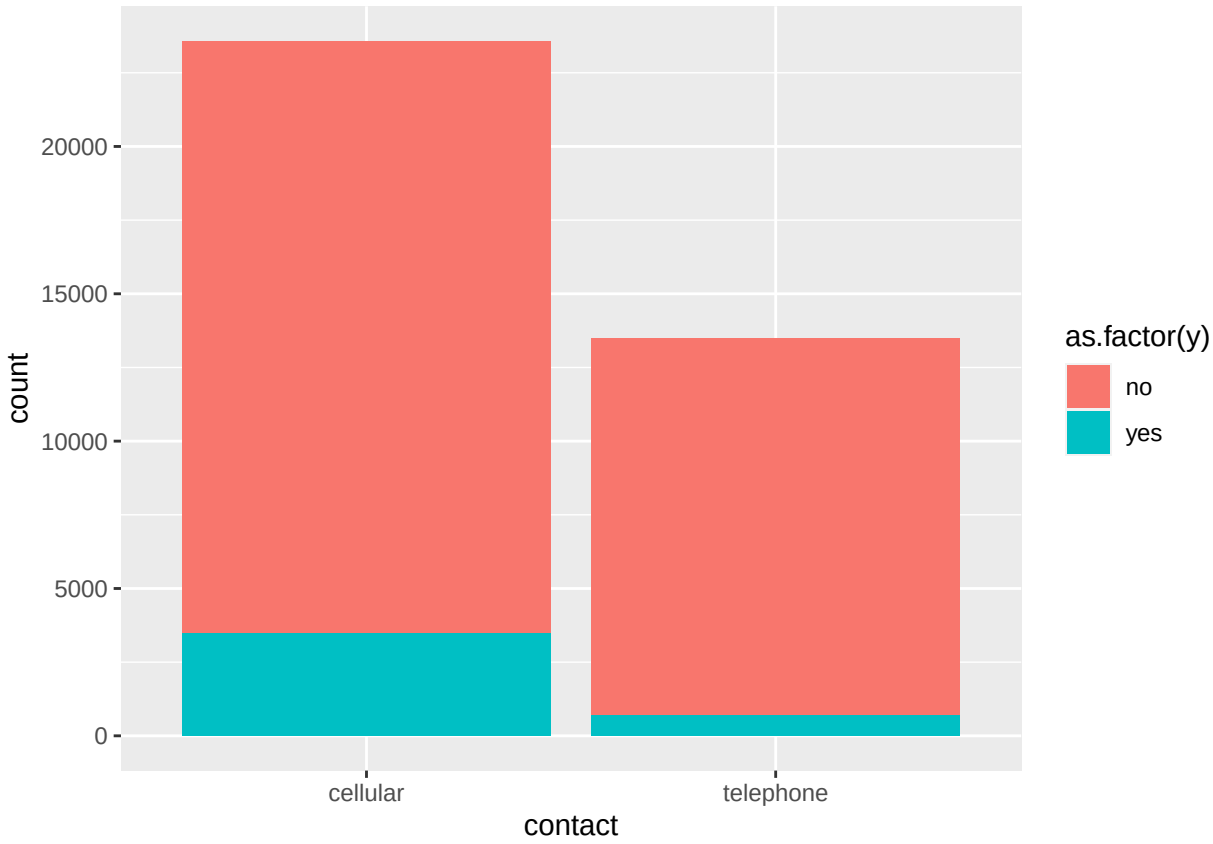
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2
##   loan    share
##   <chr>   <dbl>
## 1 no      0.113
## 2 unknown 0.110
## 3 yes     0.109
```

Nothing remarkable, we binarize.

Contact

```
calls_train %>%
  ggplot(aes(contact, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(contact) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

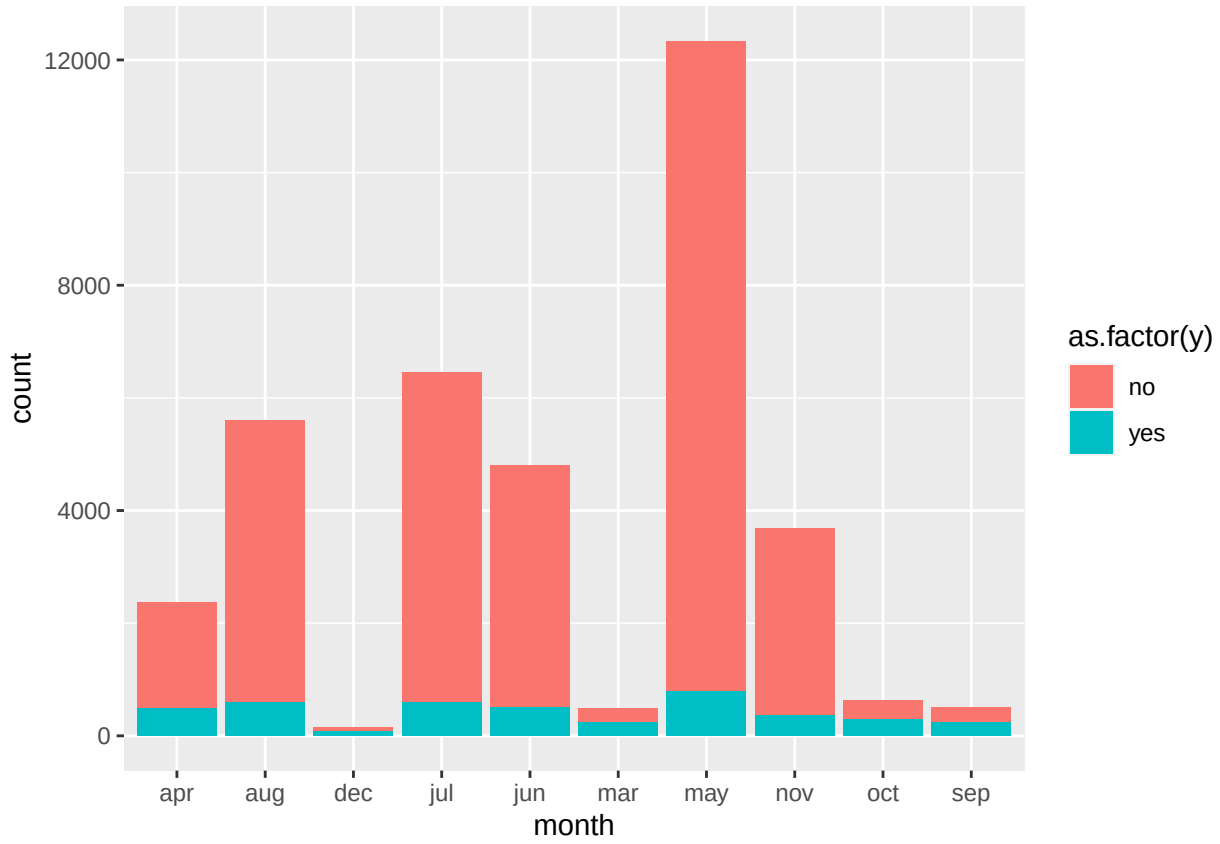
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   contact    share
##   <chr>      <dbl>
## 1 cellular  0.148
## 2 telephone 0.0512
```

Here, interestingly we see that people with cellular phones have a higher probability of subscription than people with regular phones: .148 vs .051. Therefore we will expect a binary variable cellular? yes/no to be significant.

Month

```
calls_train %>%
  ggplot(aes(month, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(month) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

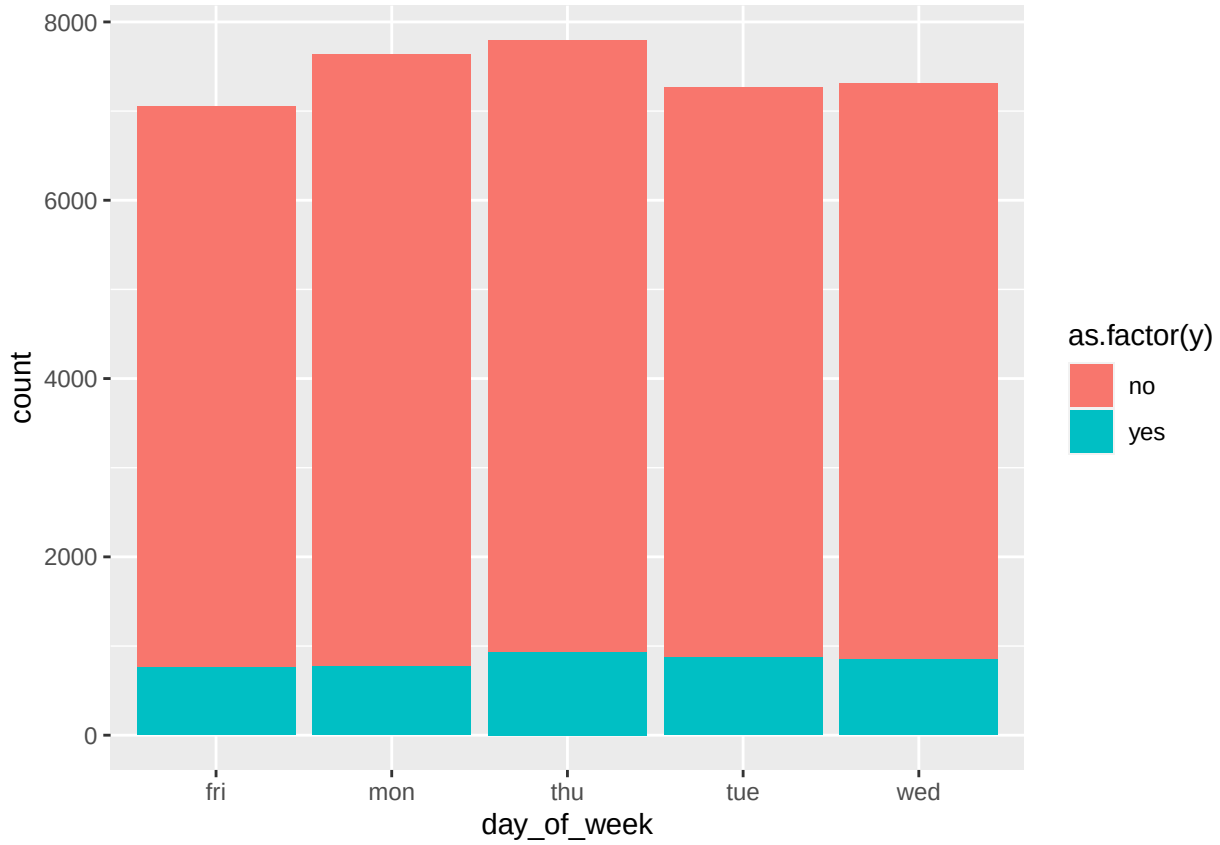
```
## # A tibble: 10 x 2
##    month  share
##    <chr>  <dbl>
##  1 apr   0.204
##  2 aug   0.106
##  3 dec   0.475
##  4 jul   0.0909
##  5 jun   0.107
##  6 mar   0.511
##  7 may   0.0639
##  8 nov   0.0981
##  9 oct   0.450
## 10 sep   0.453
```

We notice that months with small counts: March, September, October, December, see more subscriptions. It may not be significant.

Day of week

```
calls_train %>%
  ggplot(aes(day_of_week, fill=as.factor(y))) +
```

```
  geom_bar()
```




```
calls_train %>%
  group_by(day_of_week) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

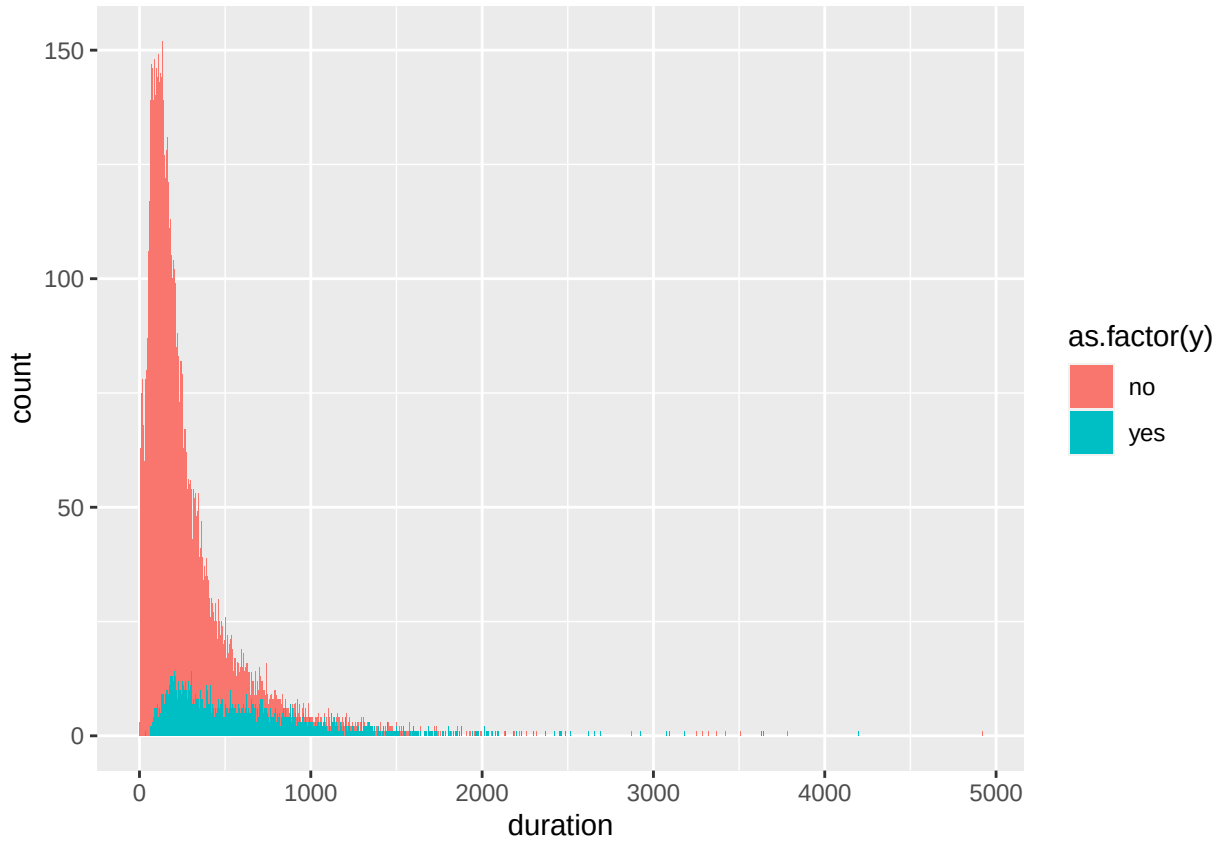
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##   day_of_week share
##   <chr>       <dbl>
## 1 fri         0.107
## 2 mon         0.101
## 3 thu         0.120
## 4 tue         0.119
## 5 wed         0.116
```

Nothing remarkable.

Duration

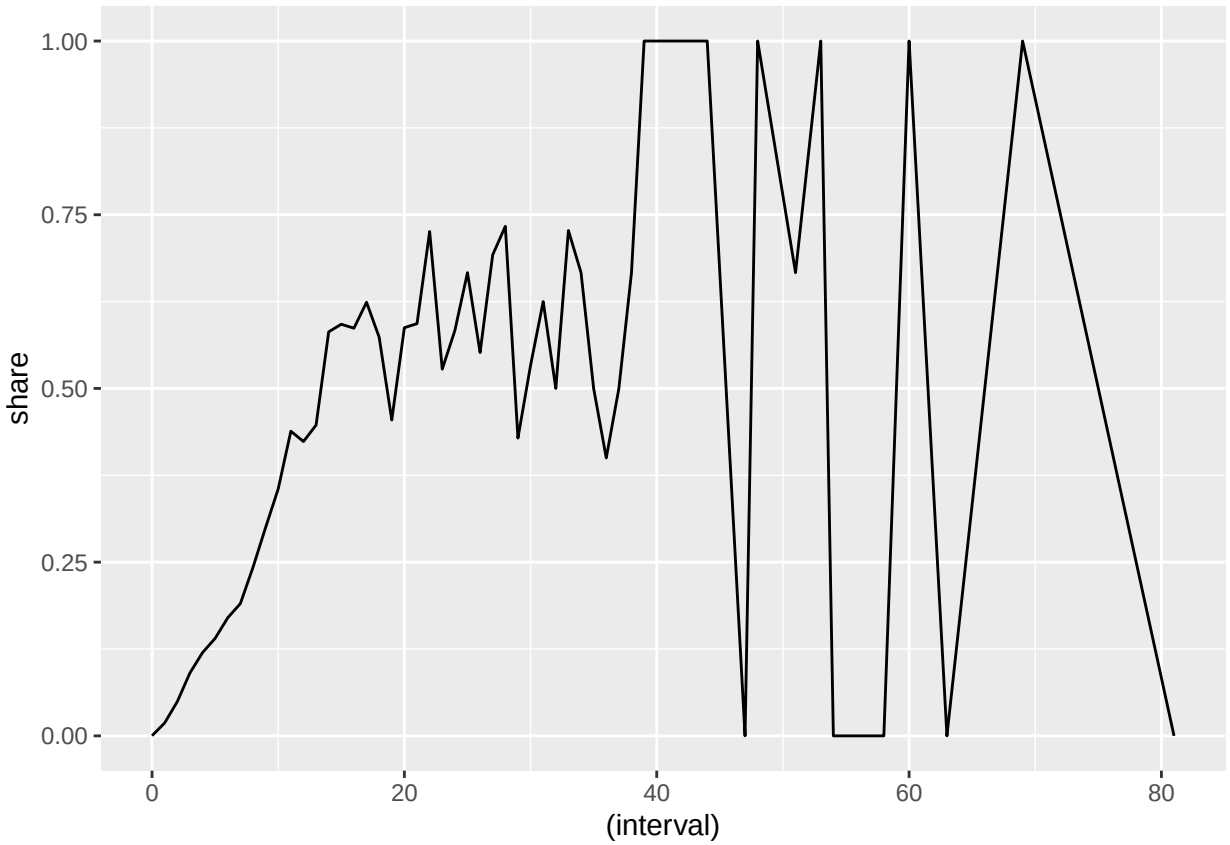
```
calls_train %>%
  ggplot(aes(duration, fill=as.factor(y))) +
  geom_bar()
```




```
duration_structure <- calls_train %>%
  mutate(interval = (duration - duration%%60)/60) %>%
  group_by(interval) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

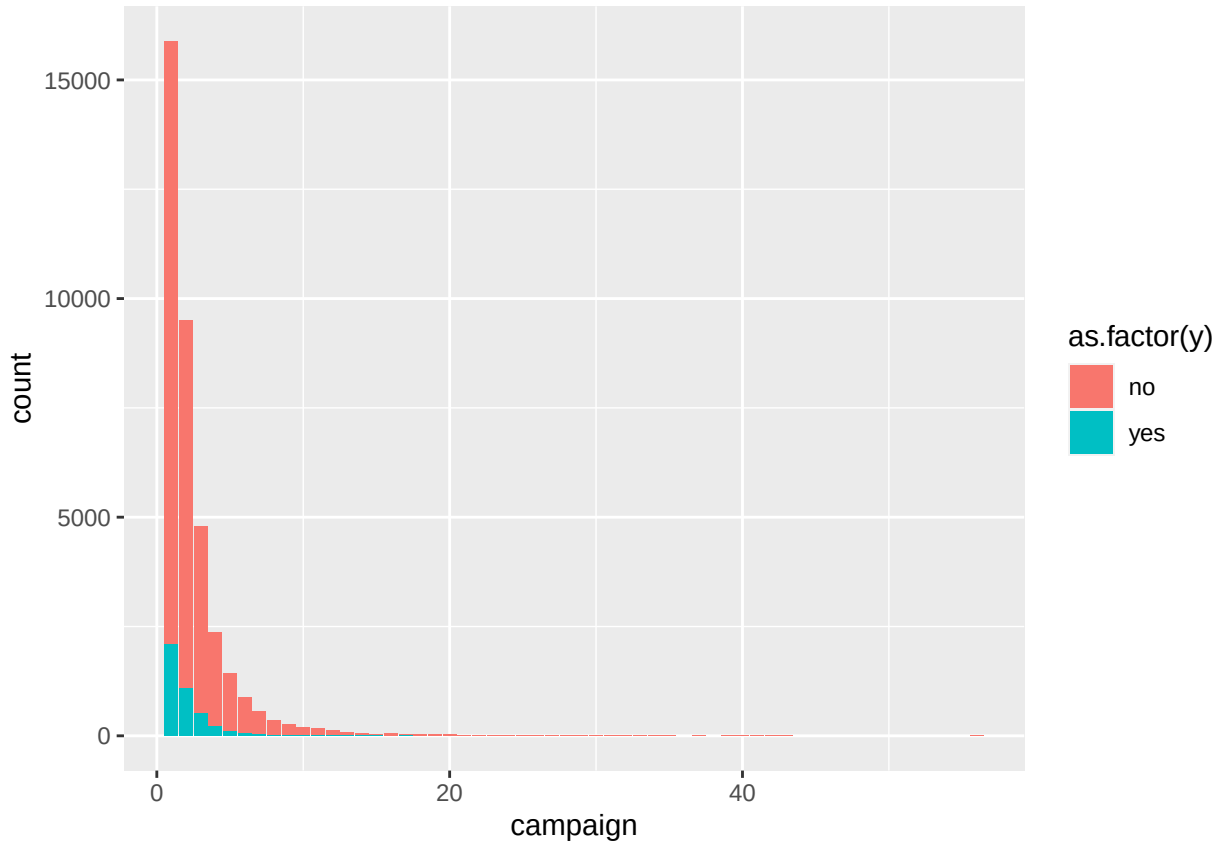
```
duration_structure %>%
  ggplot(aes(x=(interval), y=share)) + geom_line()
```



The number of values for this variable was too high so we decided to group it in minute intervals. We can see an increase of subscriptions up to 20 minutes and then the small number of values make the fluctuations hard to interpret.

Campaign

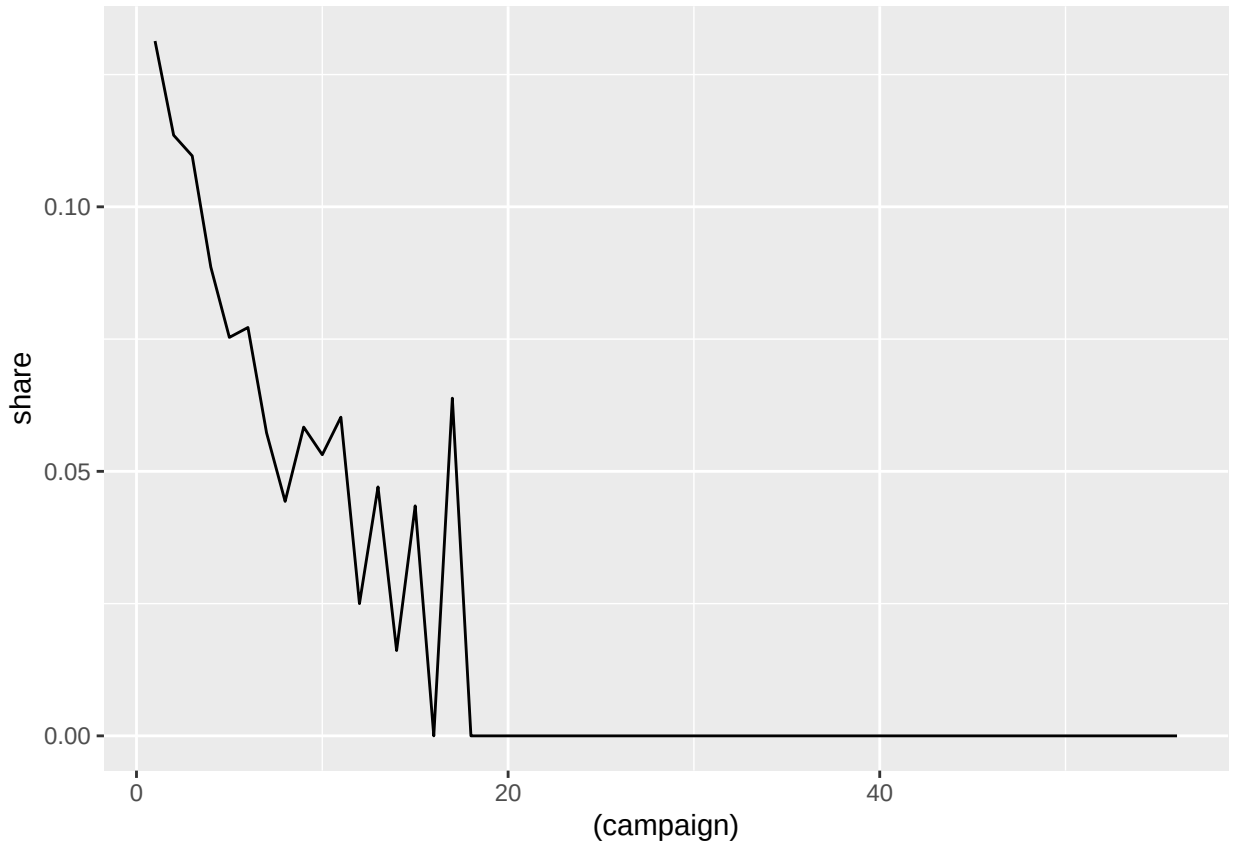
```
calls_train %>%
  ggplot(aes(campaign, fill=as.factor(y))) +
  geom_bar()
```




```
campaign_structure <- calls_train %>%
  group_by(campaign) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

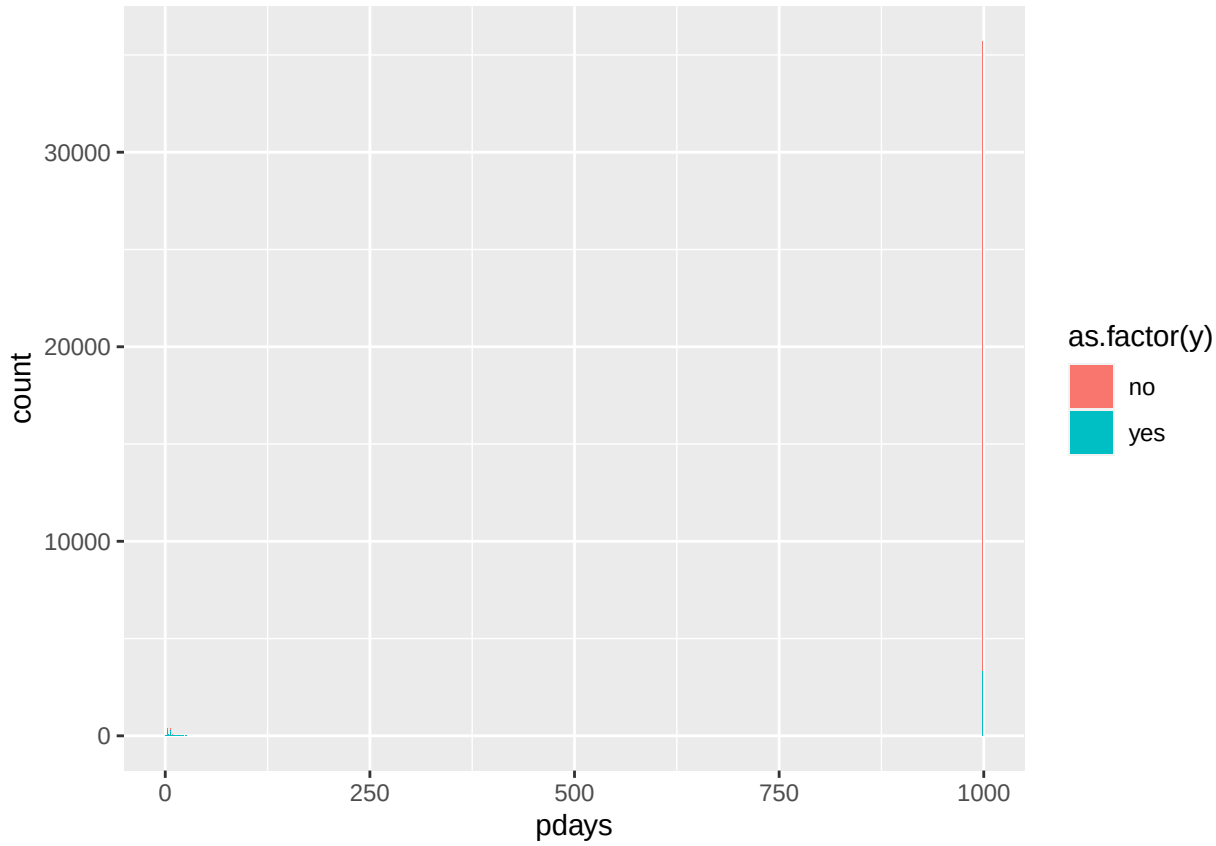
```
campaign_structure %>%
  ggplot(aes(x=(campaign), y=share)) + geom_line()
```



The subscription rate seem to decrease with the number of campaign calls.

Pdays

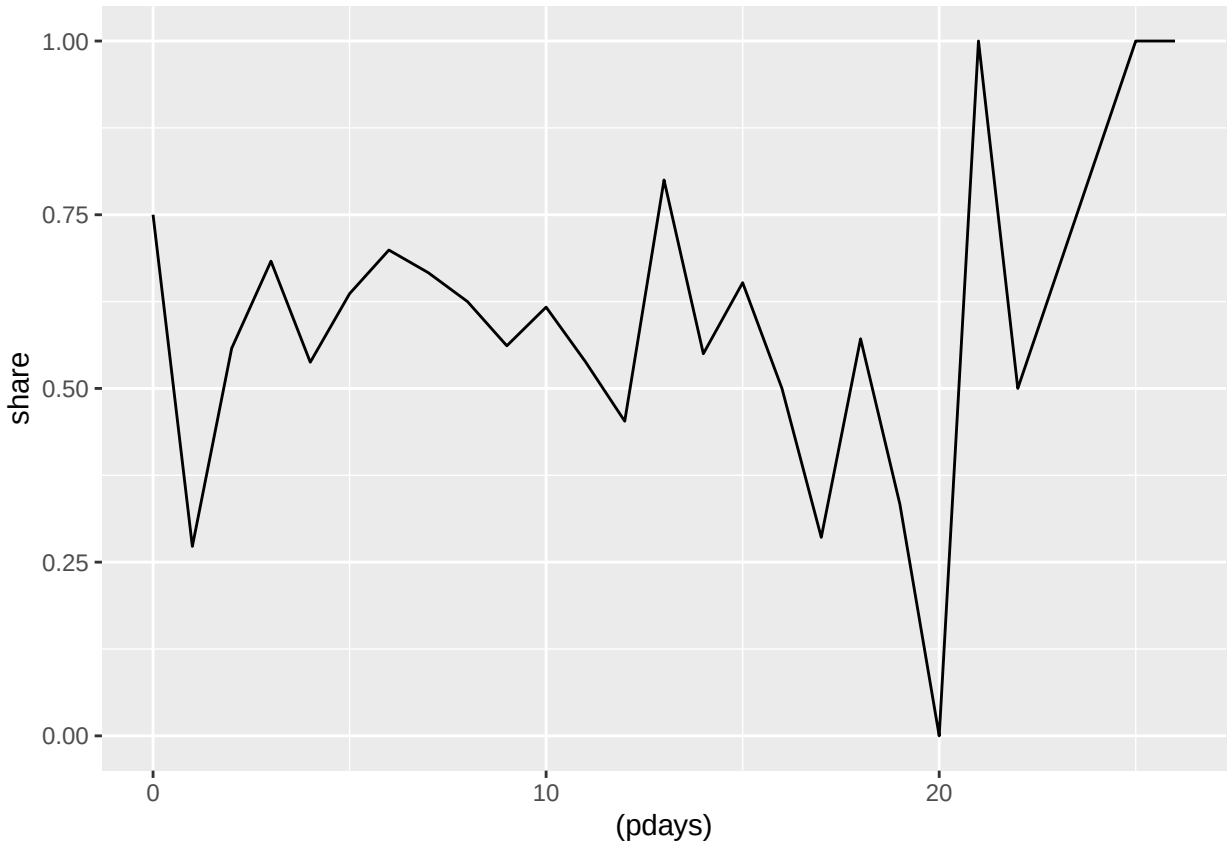
```
calls_train %>%
  ggplot(aes(pdays, fill=as.factor(y))) +
  geom_bar()
```




```
pdays_structure <- calls_train %>%
  filter(pdays<999) %>%
  group_by(pdays) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

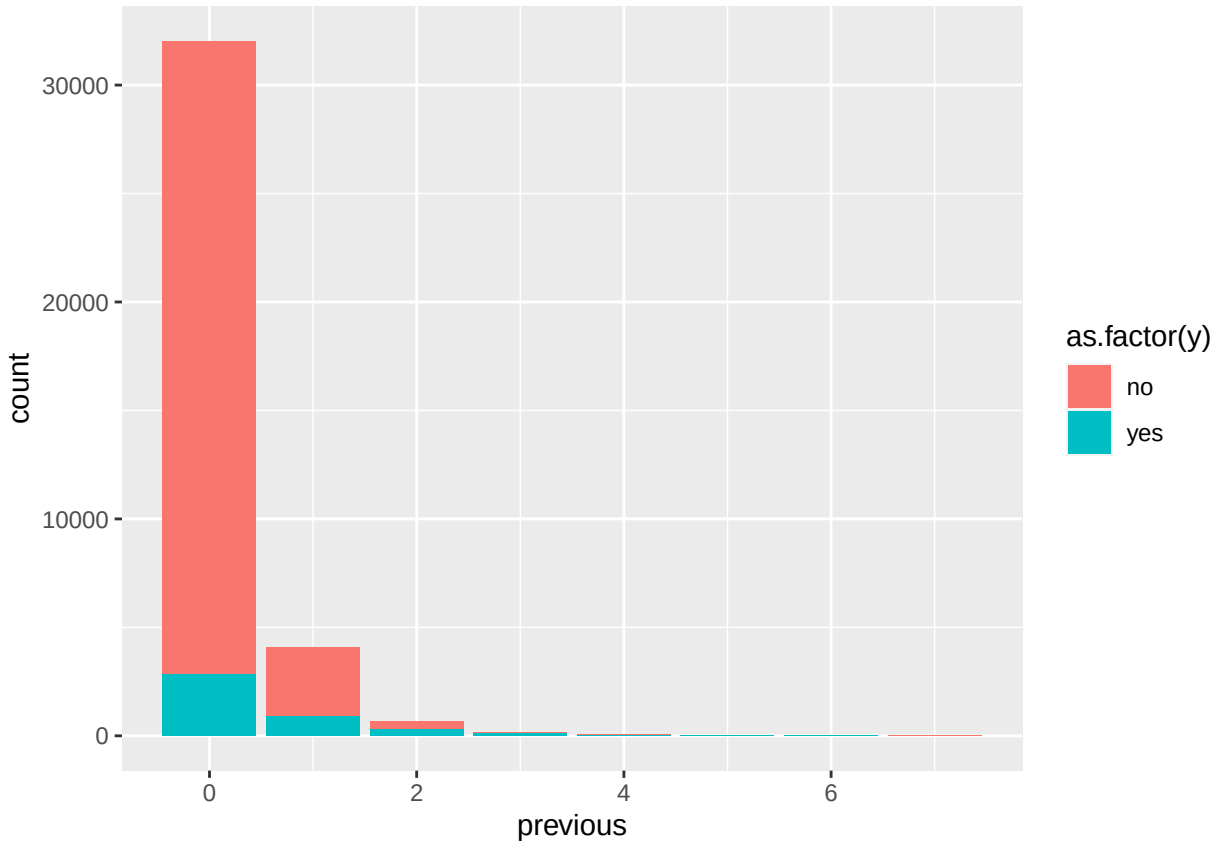
```
pdays_structure %>%
  ggplot(aes(x=(pdays), y=share)) + geom_line()
```



If we neglect the extreme value of 999, there doesn't seem to be a pattern.

Previous

```
calls_train %>%
  ggplot(aes(previous, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(previous) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

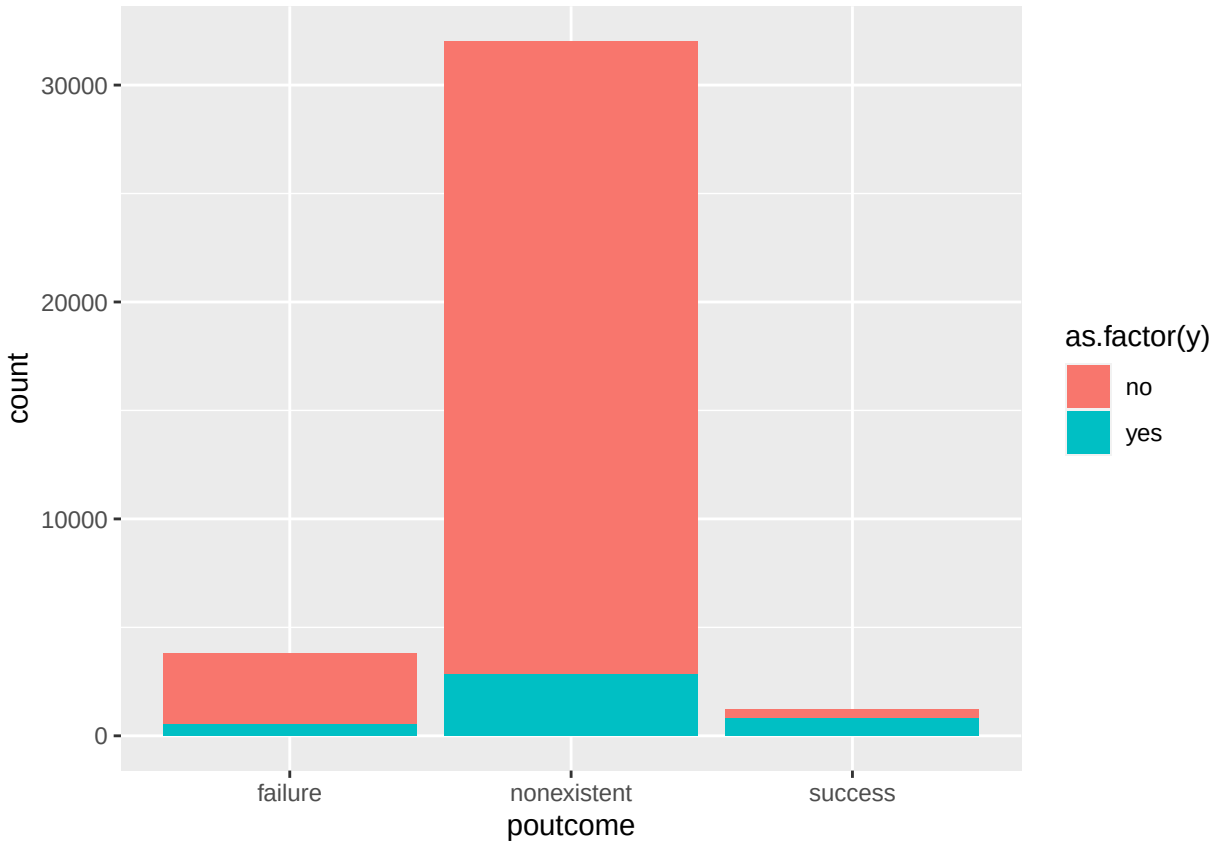
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   previous  share
##      <int>  <dbl>
## 1        0 0.0883
## 2        1 0.214
## 3        2 0.468
## 4        3 0.588
## 5        4 0.542
## 6        5 0.688
## 7        6 0.5
## 8        7 0
```

The previous number of calls before the campaign seems to matter if it differs than zero, but the counts are small.

Poutcome

```
calls_train %>%
  ggplot(aes(poutcome, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(poutcome) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

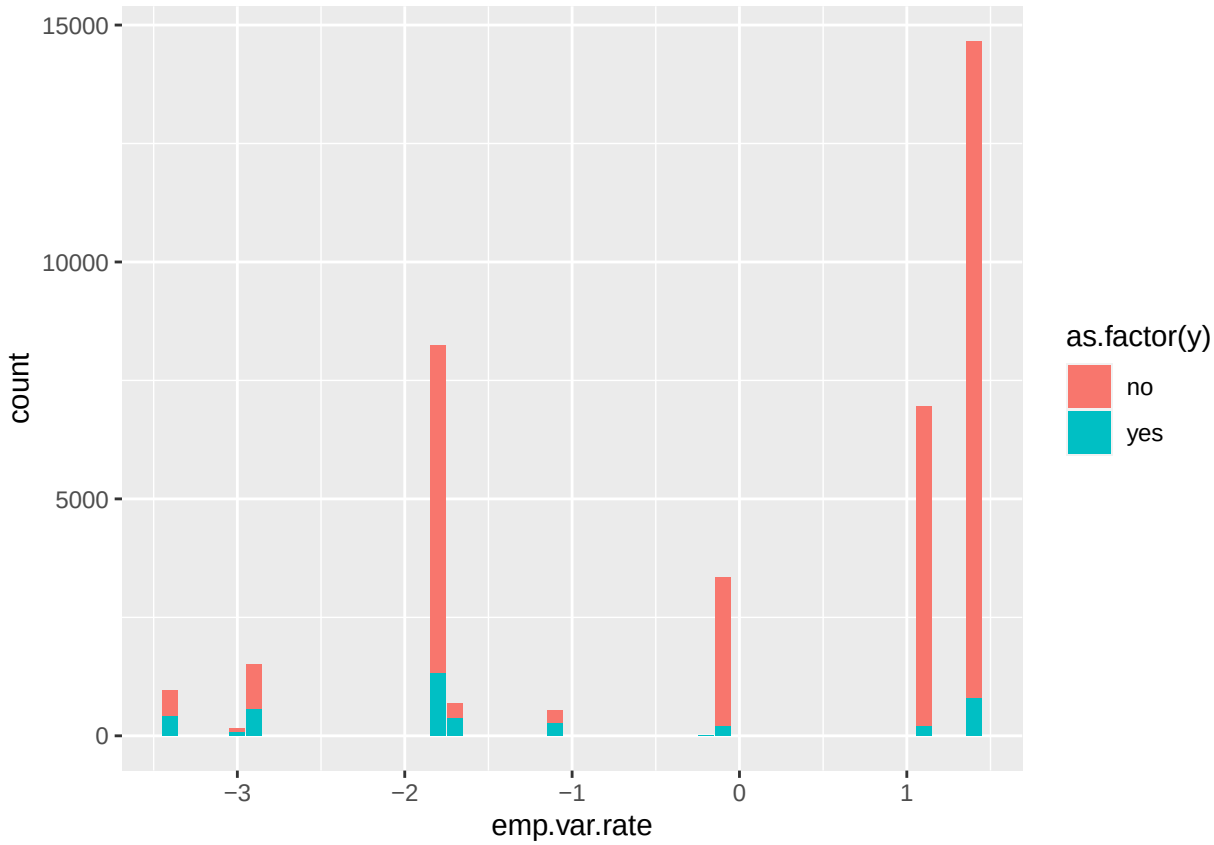
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2
##   poutcome     share
##   <chr>        <dbl>
## 1 failure     0.145
## 2 nonexistent 0.0883
## 3 success     0.651
```

We see something interesting: if a previous call has been a "success", we have a .651 probability of another subscription. Even a previous "failure" gives a better outcome than "nonexistent": .145 vs .088 .

Emp.var.rate

```
calls_train %>%
  ggplot(aes(emp.var.rate, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(emp.var.rate) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

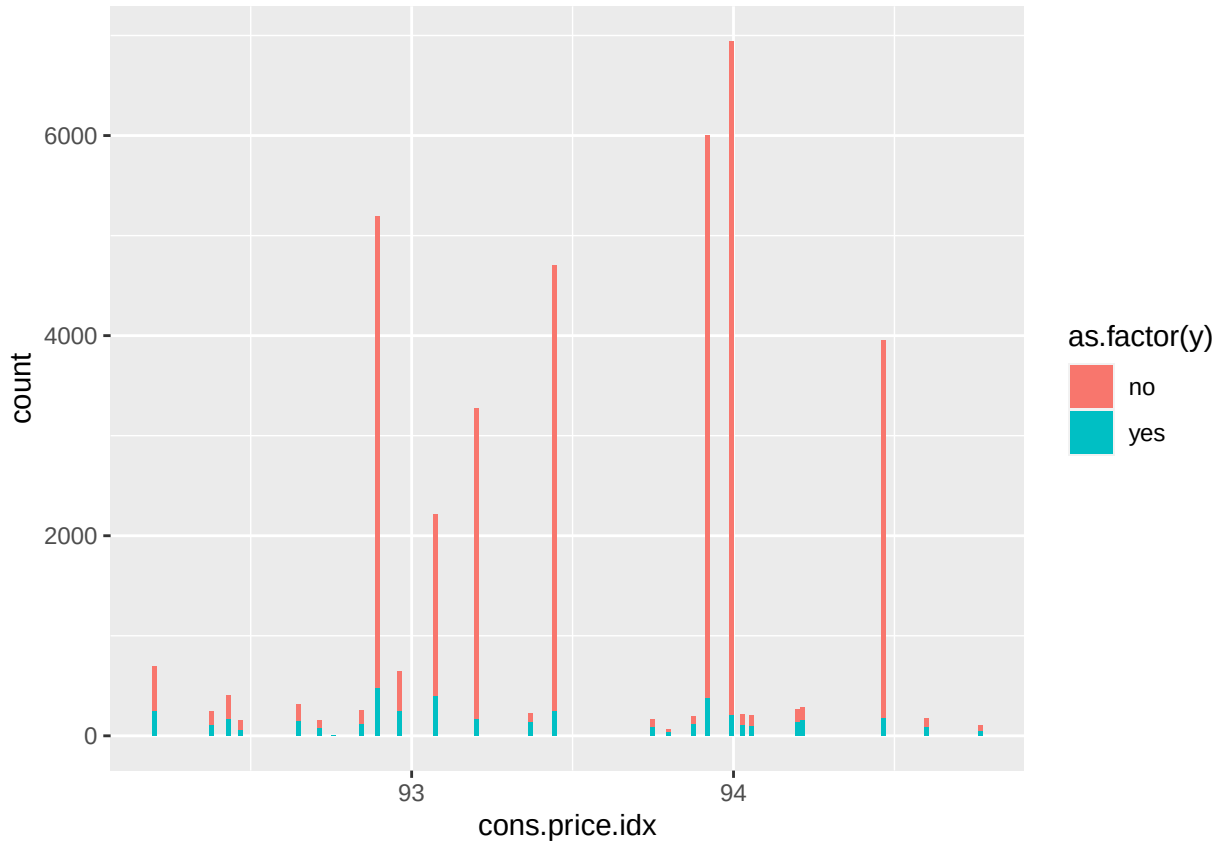
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 10 x 2
##    emp.var.rate  share
##           <dbl>  <dbl>
##  1         -3.4 0.435
##  2         -3   0.494
##  3         -2.9 0.365
##  4         -1.8 0.159
##  5         -1.7 0.519
##  6         -1.1 0.472
##  7         -0.2 0.125
##  8         -0.1 0.0618
##  9          1.1 0.0295
## 10          1.4 0.0534
```

We see that decrease in employment variation rate seems to engender more subscriptions.

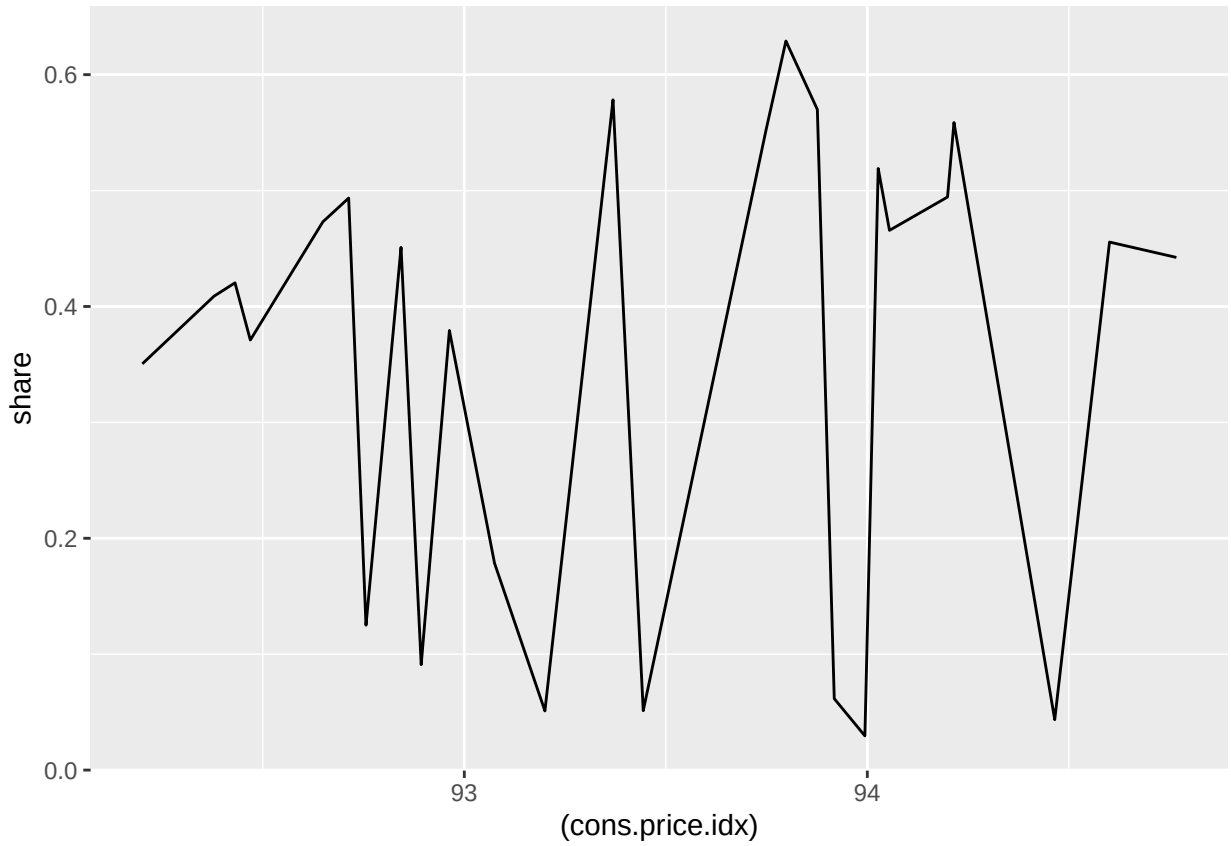
Cons.price.idx

```
calls_train %>%
  ggplot(aes(cons.price.idx, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(cons.price.idx) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no"))) %>%
  ggplot(aes(x=(cons.price.idx), y=share)) + geom_line()
```

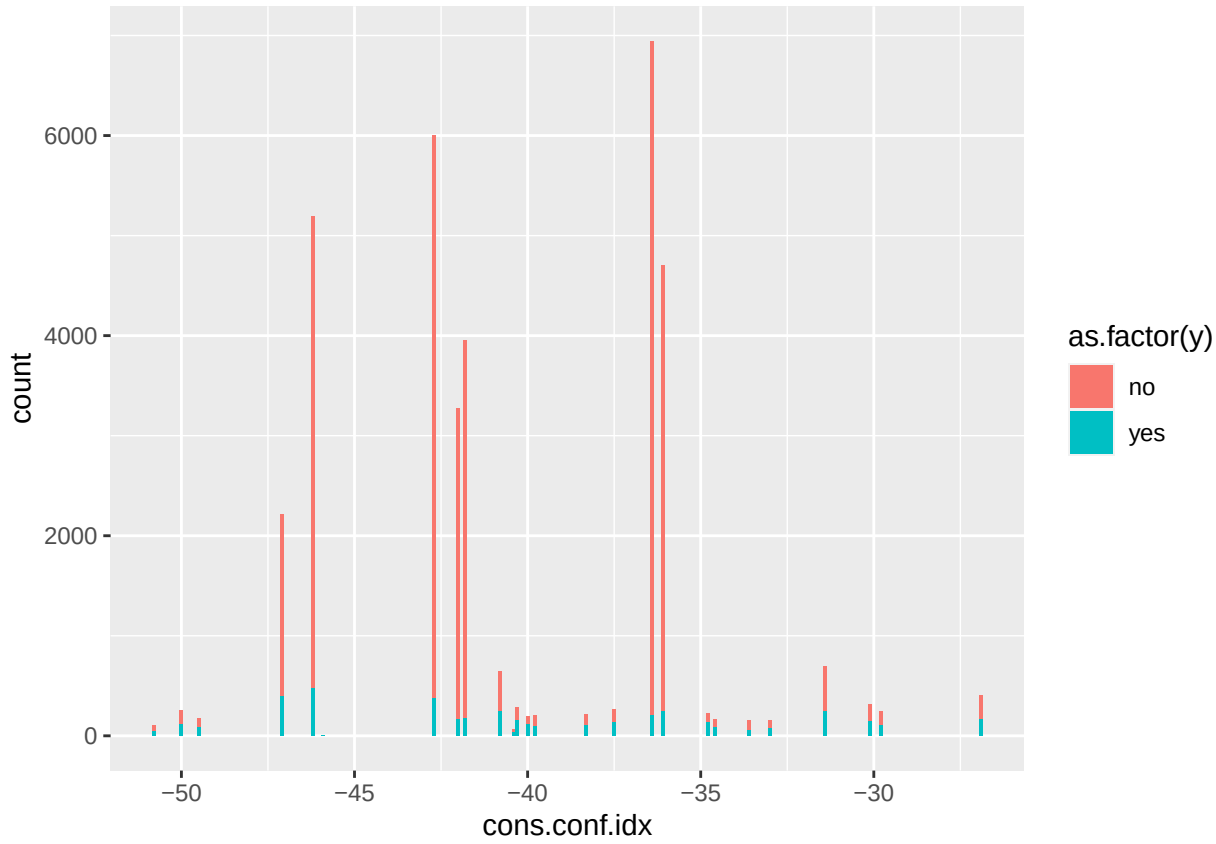
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



There is no convincing pattern here.

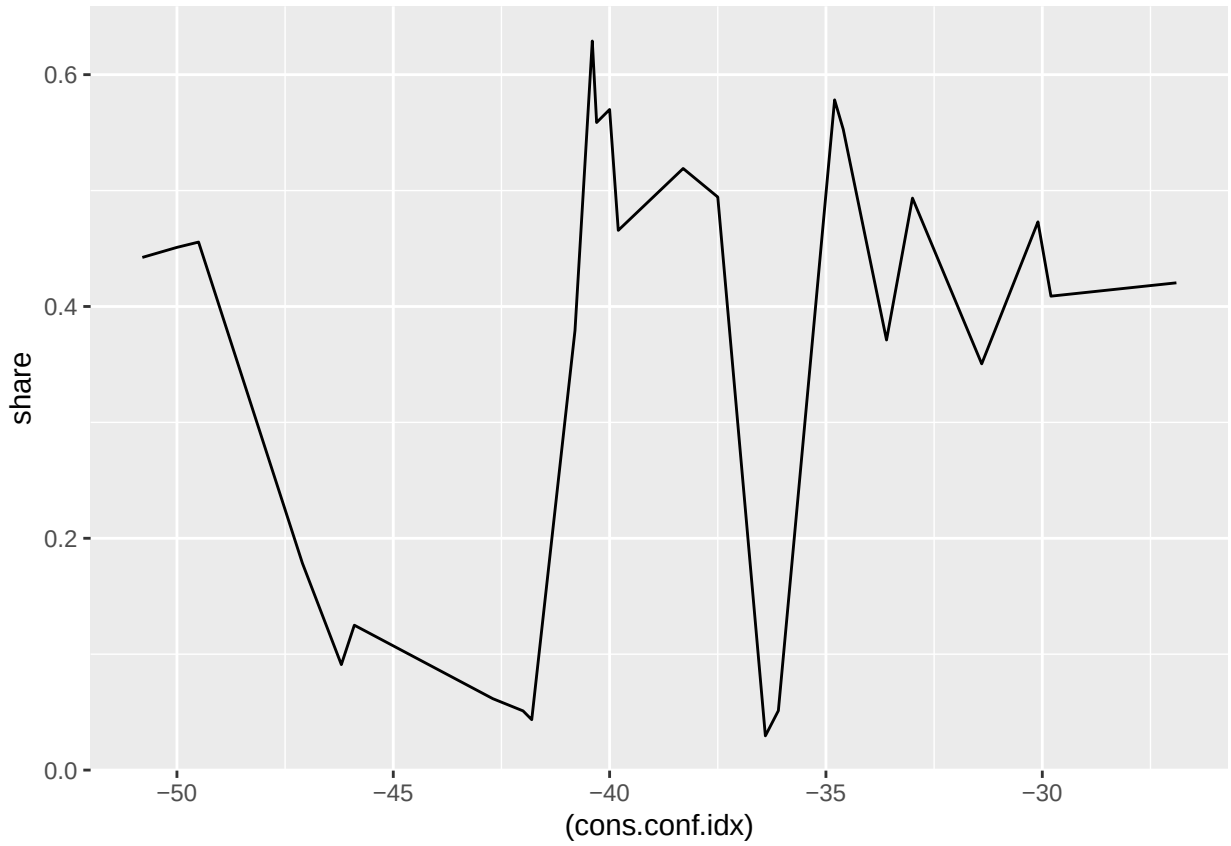
Cons.conf.idx

```
calls_train %>%
  ggplot(aes(cons.conf.idx, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(cons.conf.idx) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no"))) %>%
  ggplot(aes(x=(cons.conf.idx), y=share)) + geom_line()
```

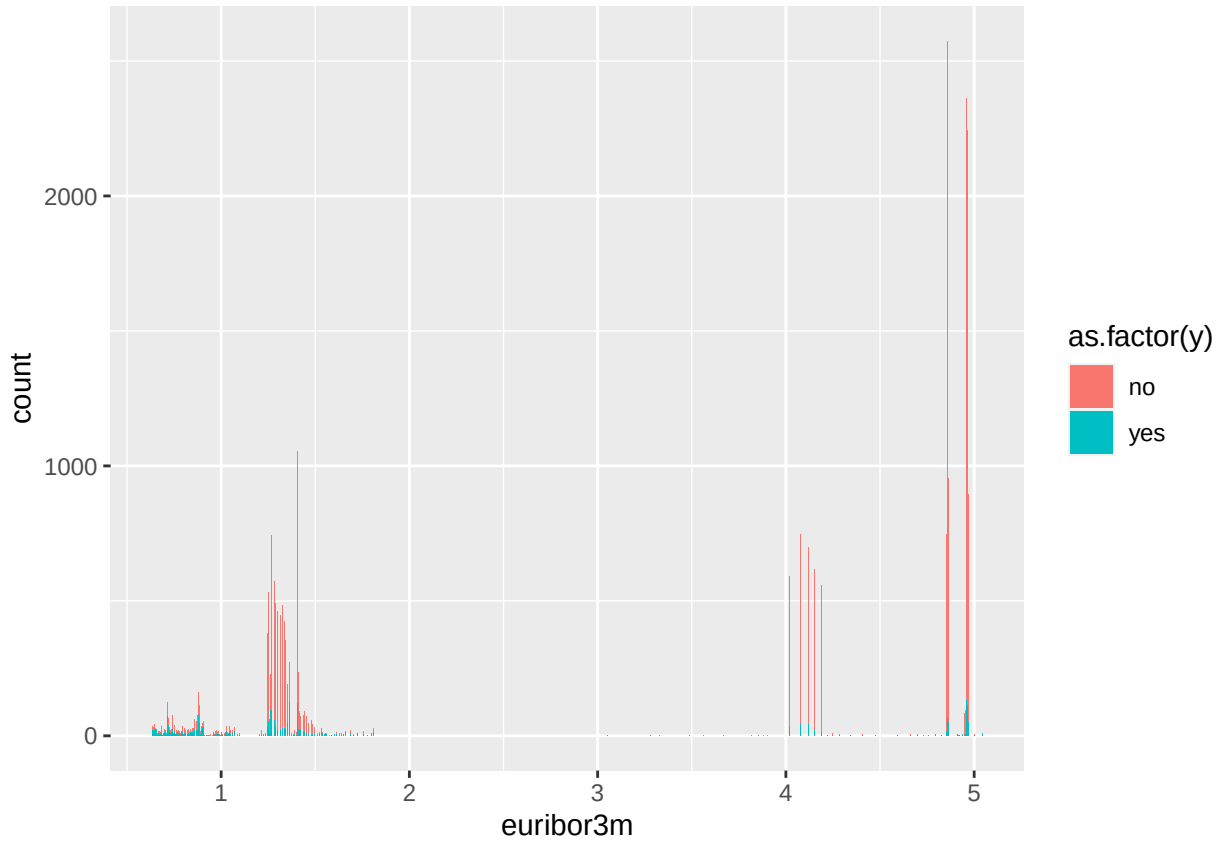
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Again, no convincing pattern.

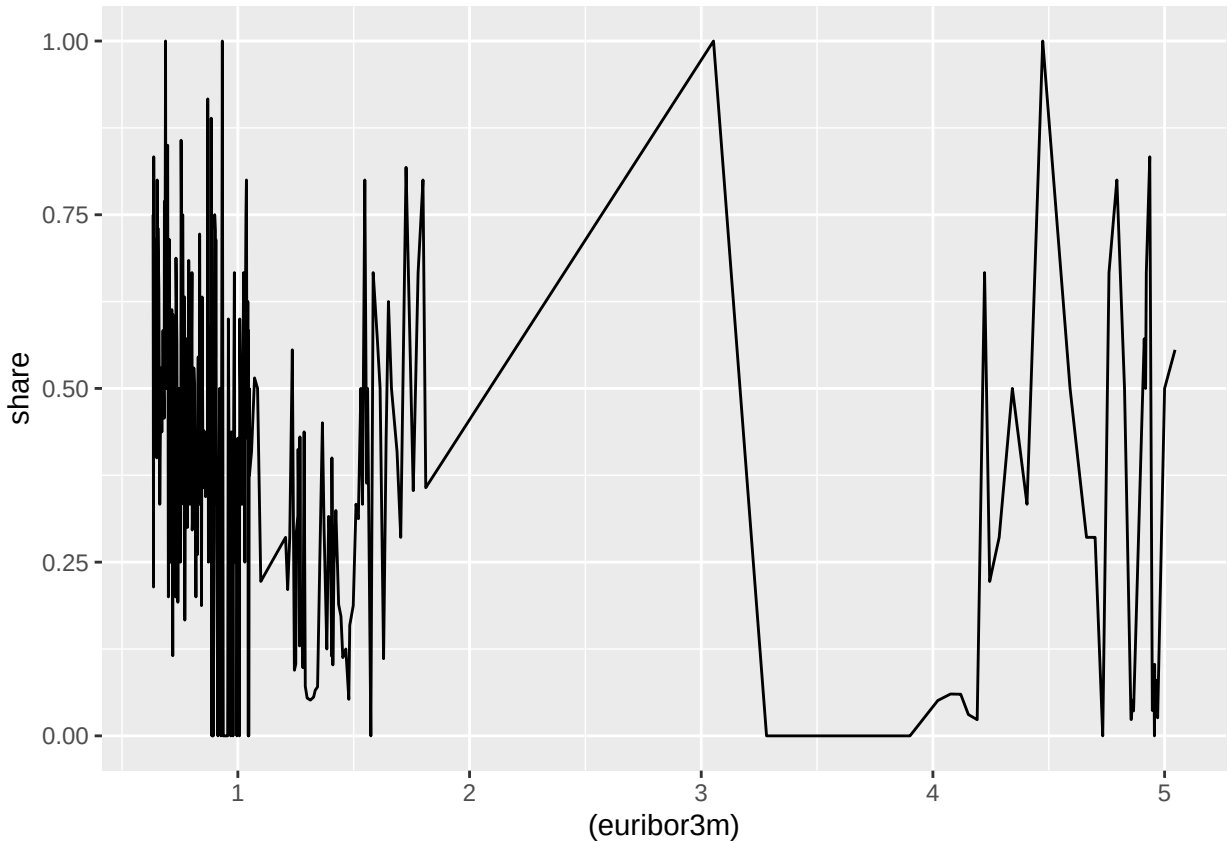
Euribor3m

```
calls_train %>%
  ggplot(aes(euribor3m, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(euribor3m) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no"))) %>%
  ggplot(aes(x=(euribor3m), y=share)) + geom_line()
```

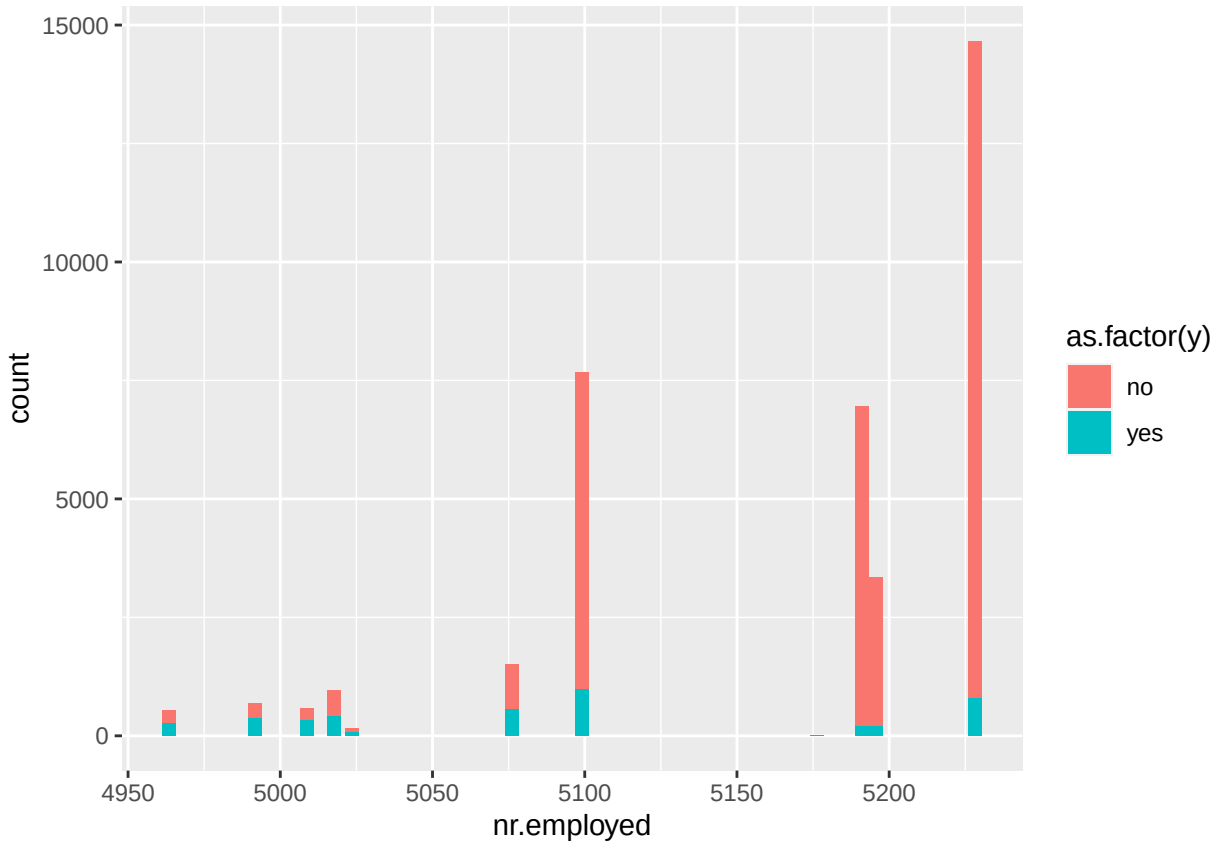
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Nothing to see here it seems.

Nr.employed

```
calls_train %>%
  ggplot(aes(nr.employed, fill=as.factor(y))) +
  geom_bar()
```




```
calls_train %>%
  group_by(nr.employed) %>%
  summarize(share = sum(y=="yes") / (sum(y=="yes")+sum(y=="no")))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 11 x 2
##    nr.employed  share
##          <dbl>  <dbl>
##  1       4964. 0.472
##  2       4992. 0.519
##  3       5009. 0.568
##  4       5018. 0.435
##  5       5024. 0.494
##  6       5076. 0.365
##  7       5099. 0.128
##  8       5176. 0.125
##  9       5191  0.0295
## 10       5196. 0.0618
## 11       5228. 0.0534
```

Nothing to see there.

Results

Linear model with the 20 variables

We train our model with a *glm* method on all variables to see what variables we would like to keep for most computing intensive methods.

```
calls_norm_train <- calls_norm[-test_index,]
calls_norm_test <- calls_norm[test_index,]
x_train <- calls_norm_train %>%
  select(-y)
y_train <- as.numeric(calls_norm_train$y)
x_test <- calls_norm_test %>%
  select(-y)
y_test <- as.numeric(calls_norm_test$y)


train_glm <- train(x_train, y_train, method="glm")
```

```
## Warning in train.default(x_train, y_train, method = "glm"): You are trying to
## do regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
predict_glm <- predict(train_glm, x_test)
predict_glm_bin <- ifelse(predict_glm >.5, 1, 0)
mean(predict_glm_bin == y_test)
```

```
## [1] 0.9101724
```

```
train_glm$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
##    (Intercept)             age             job         marital       education
##      0.0041073       0.0103968       0.0083601       0.0081145       0.0171168
##        default         housing            loan         contact           month
##     -0.0166069       0.0009889      -0.0025801       0.0930854       0.0023980
##    day_of_week        duration        campaign           pdays        previous
##      0.0017606       0.1195044       0.0028723       0.2281662      -0.0175326
##       poutcome    emp.var.rate  cons.price.idx   cons.conf.idx       euribor3m
##      0.1153419      -0.1276783       0.0897483       0.0410853       0.0001187
##    nr.employed
##     -0.0107744
##
## Degrees of Freedom: 37068 Total (i.e. Null);  37048 Residual
## Null Deviance:       3706
## Residual Deviance: 2446  AIC: 4473
```

We predict a subscription $y = 1$ when our predicted output is $> .5$. We interpret our output as the probability of subscription and therefore We predict $y = 0$ or $y = 1$ as the output is $y_hat <= .5$ or $y_hat > .5$.

Our accuracy is 91.

Linear model with 5 variables

We inspect the parameters of our previous model and see that 5 variables seem to influence the most the output. Remember that all our variables have been normalized or binarized so we expect the coefficients to reflect the importance of the parameter in question.

Our 5 variables of choice are :

- *contact* : we already predicted the importance of the variable due to the obvious difference between cellular and regular phone call.
- *duration* : again, it is expected that a longer phone call correlates with an interested customer.
- *pdays* : given that there seems to be no pattern in this variable except for the *999* value, we decided to put this value in binary and it seems to be a good predictor.
- *poutcome* : it was seen in the data that a previous *success* is highly predictive of a new one, we decided to clump *failure* and *nonexistent* together as 0 as it was not clear how to quantify a priori the difference between *success*, *failure*, and *nonexistent*.
- *emp.var.rate* : it is the most influential socio-economic parameter.

```
calls_norm_train <- calls_norm_train %>%
  select(contact, duration, pdays, poutcome, emp.var.rate,  y)
x_train <- calls_norm_train %>%
  select(-y)
y_train <- as.numeric(calls_norm_train$y)
x_test <- calls_norm_test %>%
  select(-y)
y_test <- as.numeric(calls_norm_test$y)


train_glm <- train(x_train, y_train, method="glm")
```

```
## Warning in train.default(x_train, y_train, method = "glm"): You are trying to
## do regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
predict_glm <- predict(train_glm, x_test)
predict_glm_bin <- ifelse(predict_glm >.5, 1, 0)
mean(predict_glm_bin == y_test)
```

```
## [1] 0.9053168
```

With 4 times less variables, we still get a good accuracy of 90.5. We will use these for the next model.

K nearest neighbors model

```
train_knn <- train(x_train, y_train, method="knn",
                   tuneGrid = data.frame(k = 100))
```

```
## Warning in train.default(x_train, y_train, method = "knn", tuneGrid =
## data.frame(k = 100)): You are trying to do regression and your outcome only has
## two possible values Are you trying to do classification? If so, use a 2 level
## factor as your outcome column.
```

```
predict_knn <- predict(train_knn, x_test)
predict_knn_bin <- ifelse(predict_knn >.5, 1, 0)
mean(predict_knn_bin == y_test)
```

```
## [1] 0.9058024
```

We used different k during the simulations, and end up with an accuracy of 90.6. The code is in the comments in order to shorten the compilation time.

Conclusion

With the data tools we had in hand, we have been able to perform with a good accuracy on the test set.

We detected 5 variables that seem to be relevant to whether our calls will result in a subscription.

The *knn* method did not perform better than the linear regression unfortunately.

One can wonder if, with more computing power, we could fine tuned better models to our data and give realistic predictions.