R Project

Xiao Li

2/26/2019

Data Understanding

This training dataset, provided by PetFinder.my that is a Malaysia's leading animal welfare platform, records 14993 detailed profiles of stray animals waiting for adoption in Malaysia. The data fields of this dataset are shown as below, there are 24 fields totally

```
df <- read_csv("train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   Name = col_character(),
##   RescuerID = col_character(),
##   Description = col_character(),
##   PetID = col_character(),
##   PhotoAmt = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    14993 obs. of  24 variables:
##  $ Type         : int  2 2 1 1 1 2 2 1 2 2 ...
##  $ Name         : chr  "Nibble" "No Name Yet" "Brisco" "Miko" ...
##  $ Age          : int  3 1 1 4 1 3 12 0 2 12 ...
##  $ Breed1       : int  299 265 307 307 307 266 264 307 265 265 ...
##  $ Breed2       : int  0 0 0 0 0 0 264 0 0 0 ...
##  $ Gender       : int  1 1 1 2 1 2 1 2 2 2 ...
##  $ Color1       : int  1 1 2 1 1 5 1 1 6 1 ...
##  $ Color2       : int  7 2 7 2 0 6 0 2 0 7 ...
##  $ Color3       : int  0 0 0 0 0 0 0 7 0 0 ...
##  $ MaturitySize : int  1 2 2 2 2 2 2 2 2 2 ...
##  $ FurLength    : int  1 2 2 1 1 1 3 1 2 2 ...
##  $ Vaccinated   : int  2 3 1 1 2 2 2 2 2 3 ...
##  $ Dewormed     : int  2 3 1 1 2 2 2 2 2 3 ...
##  $ Sterilized   : int  2 3 2 2 2 2 3 2 2 3 ...
##  $ Health       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Quantity     : int  1 1 1 1 1 1 1 6 1 1 ...
##  $ Fee          : int  100 0 0 150 0 0 300 0 0 0 ...
##  $ State        : int  41326 41401 41326 41401 41326 41326 41326 41326 41326 41326 ...
##  $ RescuerID    : chr  "8480853f516546f6cf33aa88cd76c379" "3082c7125d8fb66f7dd4bff4192c8b14" "fa90fa5
##  $ VideoAmt     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Description  : chr  "Nibble is a 3+ month old ball of cuteness. He is energetic and playful. I res
##  $ PetID        : chr  "86e1089a3" "6296e909a" "3422e4906" "5842f1ff5" ...
##  $ PhotoAmt     : num  1 2 7 8 3 2 3 9 6 2 ...
##  $ AdoptionSpeed: int  2 0 3 2 2 2 1 3 1 4 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 24
##   .. ..$ Type         : list()
```

```
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Name         : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ Age          : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Breed1       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Breed2       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Gender       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Color1       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Color2       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Color3       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ MaturitySize : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ FurLength    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Vaccinated   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Dewormed     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Sterilized   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Health       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Quantity     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Fee          : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ State        : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ RescuerID    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ VideoAmt     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Description  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ PetID        : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ PhotoAmt     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ AdoptionSpeed: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

Data Preparation

Since there are some variables in the dataset is useless and meaningless for predicting adoption speed, by

reading first 6 rows of the original dataset to understand each variable, I would like to remove `Name`, `PetID`, `Quantity`, `RescuerID` and `State`. Also, since this project mainly focuses on predictive analytics instead of sentiment analysis, I decided to use the dataset without including the `Description` variable.

```
head(df)
```

```
## # A tibble: 6 x 24
##    Type Name    Age Breed1 Breed2 Gender Color1 Color2 Color3 MaturitySize
##   <int> <chr> <int>  <int>  <int>  <int>  <int>  <int>  <int>        <int>
## 1     2 Nibb~     3    299      0      1      1      7      0            1
## 2     2 No N~     1    265      0      1      1      2      0            2
## 3     1 Bris~     1    307      0      1      2      7      0            2
## 4     1 Miko      4    307      0      2      1      2      0            2
## 5     1 Hunt~     1    307      0      1      1      0      0            2
## 6     2 <NA>      3    266      0      2      5      6      0            2
## # ... with 14 more variables: FurLength <int>, Vaccinated <int>,
## #   Dewormed <int>, Sterilized <int>, Health <int>, Quantity <int>,
## #   Fee <int>, State <int>, RescuerID <chr>, VideoAmt <int>,
## #   Description <chr>, PetID <chr>, PhotoAmt <dbl>, AdoptionSpeed <int>
```

By looking at the number of data at each level of animal quantity, since the `Quantity = 1` has 11565 rows that is more than other levels' data size, I would like to remove the profile data with `Quantity > 1` for focusing on analyzing the profile data with just one animal, which could make the analysis more effective and workable.

```
count(df$Quantity)
```

```
##     x  freq
## 1   1 11565
## 2   2  1422
## 3   3   726
## 4   4   531
## 5   5   333
## 6   6   185
## 7   7    84
## 8   8    52
## 9   9    33
## 10 10    19
## 11 11    10
## 12 12     6
## 13 13     2
## 14 14     2
## 15 15     4
## 16 16     3
## 17 17     3
## 18 18     1
## 19 20    12
```

```
df <- df%>%
  filter(Quantity == "1")
head(df)
```

```
## # A tibble: 6 x 24
##    Type Name    Age Breed1 Breed2 Gender Color1 Color2 Color3 MaturitySize
##   <int> <chr> <int>  <int>  <int>  <int>  <int>  <int>  <int>        <int>
## 1     2 Nibb~     3    299      0      1      1      7      0            1
## 2     2 No N~     1    265      0      1      1      2      0            2
```

```
## 3     1 Bris~     1    307      0      1      2      7      0           2
## 4     1 Miko      4    307      0      2      1      2      0           2
## 5     1 Hunt~     1    307      0      1      1      0      0           2
## 6     2 <NA>      3    266      0      2      5      6      0           2
## # ... with 14 more variables: FurLength <int>, Vaccinated <int>,
## #   Dewormed <int>, Sterilized <int>, Health <int>, Quantity <int>,
## #   Fee <int>, State <int>, RescuerID <chr>, VideoAmt <int>,
## #   Description <chr>, PetID <chr>, PhotoAmt <dbl>, AdoptionSpeed <int>
```

```
df <- subset(df, select = -c(Name, PetID, Quantity, State, RescuerID))
head(df)
```

```
## # A tibble: 6 x 19
##    Type   Age Breed1 Breed2 Gender Color1 Color2 Color3 MaturitySize
##   <int> <int>  <int>  <int>  <int>  <int>  <int>  <int>        <int>
## 1     2     3    299      0      1      1      7      0            1
## 2     2     1    265      0      1      1      2      0            2
## 3     1     1    307      0      1      2      7      0            2
## 4     1     4    307      0      2      1      2      0            2
## 5     1     1    307      0      1      1      0      0            2
## 6     2     3    266      0      2      5      6      0            2
## # ... with 10 more variables: FurLength <int>, Vaccinated <int>,
## #   Dewormed <int>, Sterilized <int>, Health <int>, Fee <int>,
## #   VideoAmt <int>, Description <chr>, PhotoAmt <dbl>, AdoptionSpeed <int>
```

Now, our final dataset for building analysis and training models has 11565 observations and 19 variables.

Word Cloud for `Description` of pets' profiles

Let's remove all non-English description first.

```
library(stringi)
Description <- stringi::stri_trans_general(df$Description, "latin-ascii")
```

Then, clean our `Description`.

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
description<-Corpus(VectorSource(Description))
```

```
description <- tm_map(description,stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(description, stripWhitespace):
## transformation drops documents
```

```
description <- tm_map(description,tolower)
```

```
## Warning in tm_map.SimpleCorpus(description, tolower): transformation drops
## documents
```

```
description <- tm_map(description,removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(description, removeNumbers): transformation
## drops documents
```

```
description <- tm_map(description,removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(description, removePunctuation):
## transformation drops documents
```

```
description <- tm_map(description,removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(description, removeWords,
## stopwords("english")): transformation drops documents
```

```
description <- tm_map(description, removeWords,
  c("and","the","our","that","for","are","also","more","has","must","have","should","this","with","dog"
```

```
## Warning in tm_map.SimpleCorpus(description, removeWords, c("and", "the", :
## transformation drops documents
```

```
tdm_desc <- TermDocumentMatrix (description) #Creates a TDM
TDM <- as.matrix(tdm_desc) #Convert this into a matrix format
v <- sort(rowSums(TDM), decreasing = TRUE) #Gives you the frequencies for every word
summary(v)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00   16.86    5.00 4260.00
```

Now, we can create the colorful word cloud to get a glance of those frequently used terms how these animals described in their profiles.

```
wordcloud (description, scale=c(3.5,0.5), max.words=100, random.order=FALSE, rot.per=0.35, use.r.layout=
```

Right now, let's remove the `Description` from original dataset for the following general statistical analysis.

```
df1 <- subset(df, select = -c(Description))
```

Data Audit

```
cor(df1)
```

```
##                      Type         Age       Breed1        Breed2
## Type          1.000000000 -0.15826854  0.093477660 -0.0604497351
## Age          -0.158268536  1.00000000 -0.321780482 -0.0530920198
## Breed1        0.093477660 -0.32178048  1.000000000 -0.1815777331
## Breed2       -0.060449735 -0.05309202 -0.181577733  1.0000000000
## Gender       -0.037594876 -0.04279073  0.036952983  0.0096553725
## Color1        0.105645072  0.07283050 -0.020394946 -0.0167266770
## Color2        0.237447354 -0.03336170 -0.008255226  0.0049095534
## Color3        0.185815279 -0.00047442 -0.037324941  0.0218876521
## MaturitySize -0.148714187  0.07176099 -0.016274253  0.0657404731
## FurLength    -0.010202574  0.15427909 -0.110417046  0.1099348967
## Vaccinated    0.106384664 -0.11387498  0.031184533 -0.0001557164
## Dewormed      0.019402980 -0.02273108 -0.009864371 -0.0228795451
## Sterilized    0.014024255 -0.17020018  0.042879777 -0.0200631398
## Health       -0.003102668  0.10938827 -0.036837971 -0.0392012645
## Fee          -0.040014273  0.08542003 -0.196803168  0.0182276821
## VideoAmt      0.008329142 -0.01913310  0.021799312 -0.0065294514
## PhotoAmt      0.065629270 -0.06450558  0.025289601  0.0414575470
```

```
## AdoptionSpeed -0.094054004  0.11353334  0.107186533 -0.0274042664
##                     Gender       Color1       Color2        Color3
## Type          -0.037594876  0.105645072  0.237447354  1.858153e-01
## Age           -0.042790734  0.072830501 -0.033361703 -4.744200e-04
## Breed1         0.036952983 -0.020394946 -0.008255226 -3.732494e-02
## Breed2         0.009655372 -0.016726677  0.004909553  2.188765e-02
## Gender         1.000000000 -0.031704706 -0.027701374  8.877852e-02
## Color1        -0.031704706  1.000000000 -0.131610702 -2.469889e-01
## Color2        -0.027701374 -0.131610702  1.000000000  9.150047e-02
## Color3         0.088778523 -0.246988927  0.091500470  1.000000e+00
## MaturitySize  -0.053550580 -0.044707931 -0.059221919 -2.606885e-02
## FurLength     -0.015040562  0.070438043 -0.016357918  3.597295e-02
## Vaccinated    -0.016471682  0.006277476  0.024747234  1.259970e-02
## Dewormed      -0.019186250  0.008973591  0.004070577 -7.364919e-06
## Sterilized    -0.064101041 -0.016967675  0.007557271 -3.069440e-03
## Health        -0.035066279  0.024114360 -0.005193567 -1.517155e-02
## Fee           -0.013769236  0.051314599 -0.019852341  1.806714e-03
## VideoAmt       0.012012383 -0.001926632  0.023558146  1.643091e-02
## PhotoAmt       0.006836378 -0.009326108  0.066517078  4.357641e-02
## AdoptionSpeed  0.062976798 -0.036855980 -0.045265438 -1.955836e-02
##                MaturitySize    FurLength    Vaccinated      Dewormed
## Type          -0.148714187 -0.010202574  0.1063846641  1.940298e-02
## Age            0.071760992  0.154279091 -0.1138749820 -2.273108e-02
## Breed1        -0.016274253 -0.110417046  0.0311845325 -9.864371e-03
## Breed2         0.065740473  0.109934897 -0.0001557164 -2.287955e-02
## Gender        -0.053550580 -0.015040562 -0.0164716821 -1.918625e-02
## Color1        -0.044707931  0.070438043  0.0062774757  8.973591e-03
## Color2        -0.059221919 -0.016357918  0.0247472342  4.070577e-03
## Color3        -0.026068848  0.035972946  0.0125996981 -7.364919e-06
## MaturitySize   1.000000000  0.092973597 -0.0833434571 -5.518213e-02
## FurLength      0.092973597  1.000000000  0.0023605291  2.109417e-02
## Vaccinated    -0.083343457  0.002360529  1.0000000000  7.188604e-01
## Dewormed      -0.055182134  0.021094172  0.7188604361  1.000000e+00
## Sterilized    -0.061498411  0.048289288  0.4618431297  4.122610e-01
## Health        -0.015416085  0.032822222  0.0847813553  8.149851e-02
## Fee            0.036220841  0.168094372 -0.1220240355 -1.026625e-01
## VideoAmt       0.003080387 -0.009771306 -0.0238788014 -3.079135e-02
## PhotoAmt       0.012582508 -0.028991007 -0.0691044093 -1.235516e-01
## AdoptionSpeed  0.047385381 -0.102419282 -0.0700294940 -1.793969e-02
##                 Sterilized       Health          Fee     VideoAmt
## Type           0.014024255 -0.003102668 -0.040014273  0.008329142
## Age           -0.170200176  0.109388272  0.085420032 -0.019133097
## Breed1         0.042879777 -0.036837971 -0.196803168  0.021799312
## Breed2        -0.020063140 -0.039201264  0.018227682 -0.006529451
## Gender        -0.064101041 -0.035066279 -0.013769236  0.012012383
## Color1        -0.016967675  0.024114360  0.051314599 -0.001926632
## Color2         0.007557271 -0.005193567 -0.019852341  0.023558146
## Color3        -0.003069440 -0.015171546  0.001806714  0.016430906
## MaturitySize  -0.061498411 -0.015416085  0.036220841  0.003080387
## FurLength      0.048289288  0.032822222  0.168094372 -0.009771306
## Vaccinated     0.461843130  0.084781355 -0.122024036 -0.023878801
## Dewormed       0.412260988  0.081498510 -0.102662489 -0.030791352
## Sterilized     1.000000000  0.062719836 -0.065869651 -0.014635868
## Health         0.062719836  1.000000000 -0.017523423 -0.004308715
```

```
## Fee           -0.065869651 -0.017523423  1.000000000  0.002745967
## VideoAmt      -0.014635868 -0.004308715  0.002745967  1.000000000
## PhotoAmt      -0.085700694 -0.016440643  0.019574640  0.225334052
## AdoptionSpeed -0.109845272  0.044111251 -0.007290371 -0.005806922
##                   PhotoAmt AdoptionSpeed
## Type           0.065629270  -0.094054004
## Age           -0.064505583   0.113533339
## Breed1         0.025289601   0.107186533
## Breed2         0.041457547  -0.027404266
## Gender         0.006836378   0.062976798
## Color1        -0.009326108  -0.036855980
## Color2         0.066517078  -0.045265438
## Color3         0.043576413  -0.019558359
## MaturitySize   0.012582508   0.047385381
## FurLength     -0.028991007  -0.102419282
## Vaccinated    -0.069104409  -0.070029494
## Dewormed      -0.123551567  -0.017939689
## Sterilized    -0.085700694  -0.109845272
## Health        -0.016440643   0.044111251
## Fee            0.019574640  -0.007290371
## VideoAmt       0.225334052  -0.005806922
## PhotoAmt       1.000000000  -0.022720966
## AdoptionSpeed -0.022720966   1.000000000
```

```
str(df1)
```

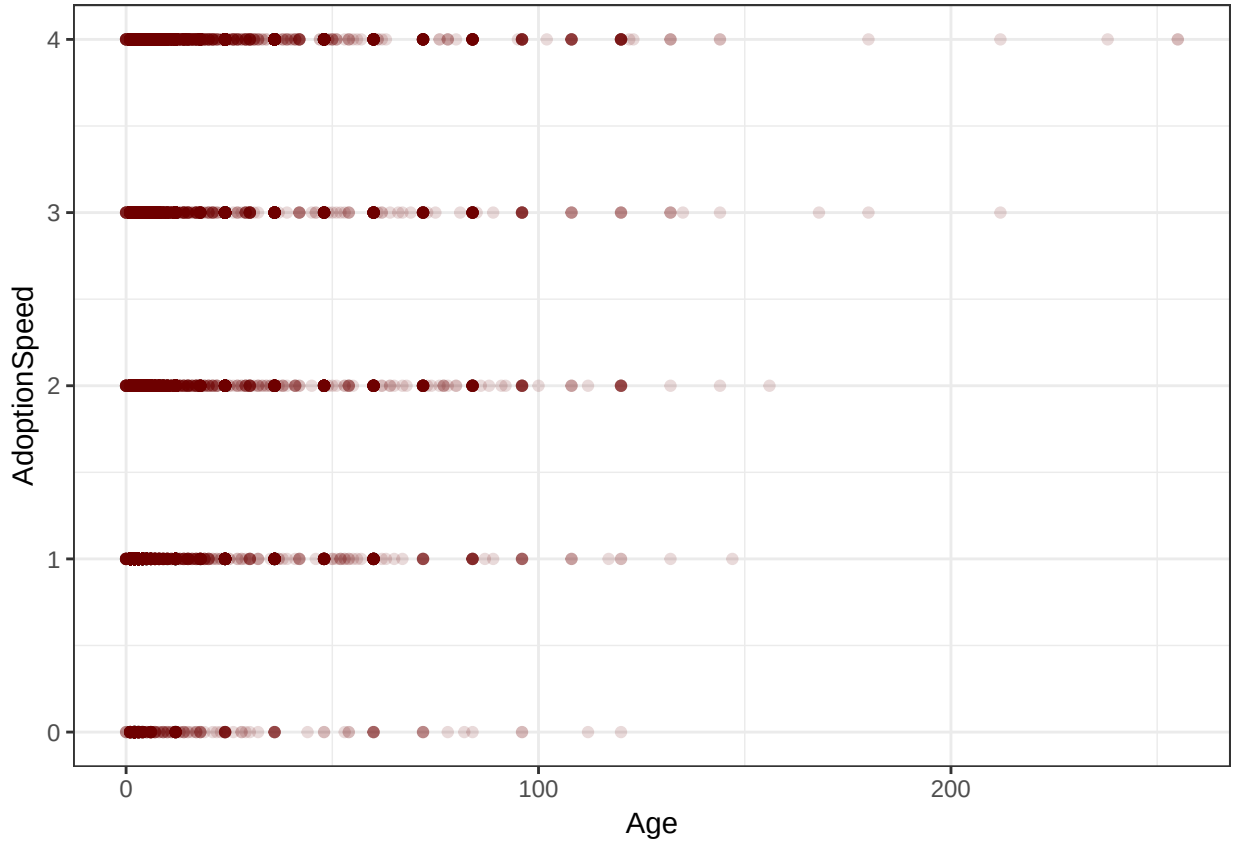
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    11565 obs. of  18 variables:
##  $ Type         : int  2 2 1 1 1 2 2 2 2 1 ...
##  $ Age          : int  3 1 1 4 1 3 12 2 12 2 ...
##  $ Breed1       : int  299 265 307 307 307 266 264 265 265 307 ...
##  $ Breed2       : int  0 0 0 0 0 0 264 0 0 0 ...
##  $ Gender       : int  1 1 1 2 1 2 1 2 2 1 ...
##  $ Color1       : int  1 1 2 1 1 5 1 6 1 1 ...
##  $ Color2       : int  7 2 7 2 0 6 0 0 7 2 ...
##  $ Color3       : int  0 0 0 0 0 0 0 0 0 7 ...
##  $ MaturitySize : int  1 2 2 2 2 2 2 2 2 2 ...
##  $ FurLength    : int  1 2 2 1 1 1 3 2 2 1 ...
##  $ Vaccinated   : int  2 3 1 1 2 2 2 2 3 2 ...
##  $ Dewormed     : int  2 3 1 1 2 2 2 2 3 1 ...
##  $ Sterilized   : int  2 3 2 2 2 2 3 2 3 2 ...
##  $ Health       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Fee          : int  100 0 0 150 0 0 300 0 0 0 ...
##  $ VideoAmt     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PhotoAmt     : num  1 2 7 8 3 2 3 6 2 7 ...
##  $ AdoptionSpeed: int  2 0 3 2 2 2 1 1 4 1 ...
```

continuous variables vs. target

Let's plot the relationships between continuous variables and target variable, `AdoptionSpeed`.

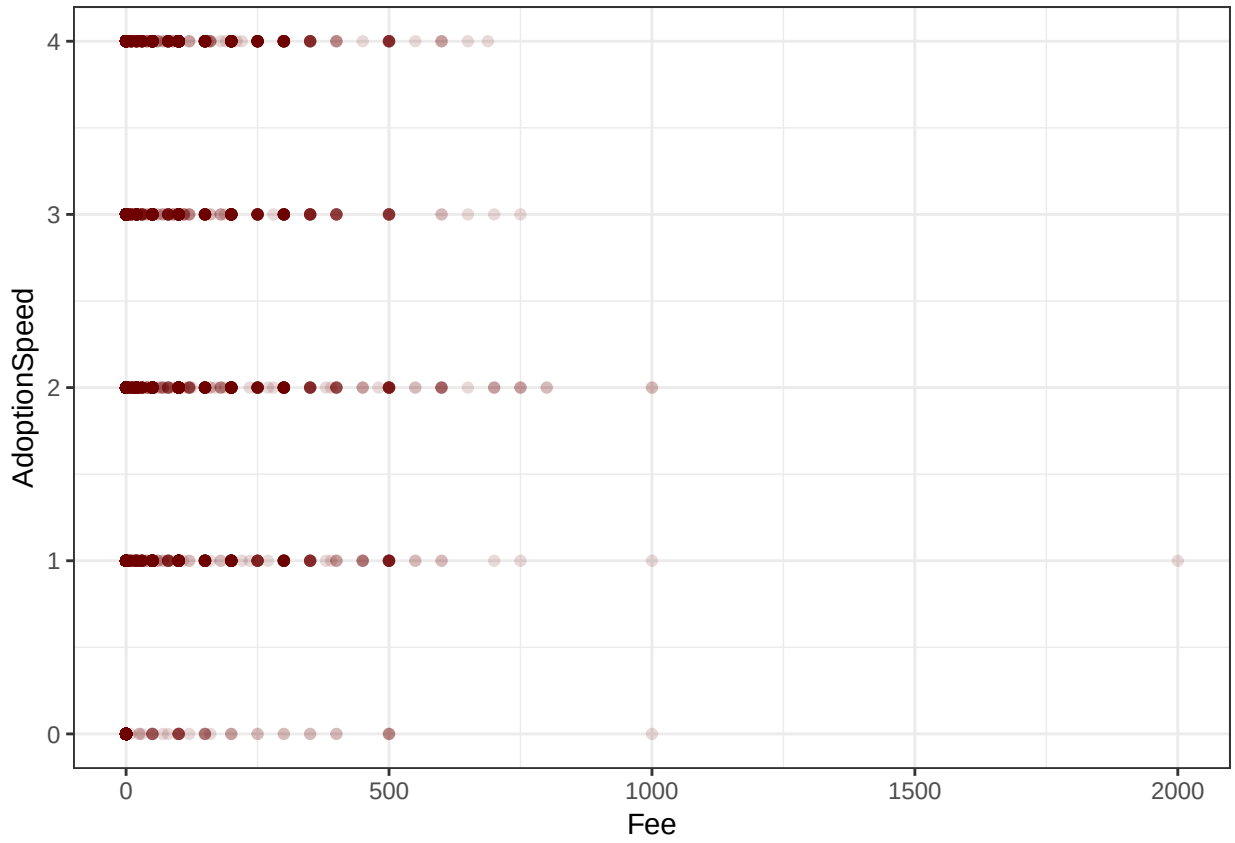
- `Age(in months)` vs. `AdoptionSpeed`: by reading the scatter plot, it's obviously to see that younger animals would be more quickly to be adopted.

```
p <- ggplot(df1, aes(x = Age, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



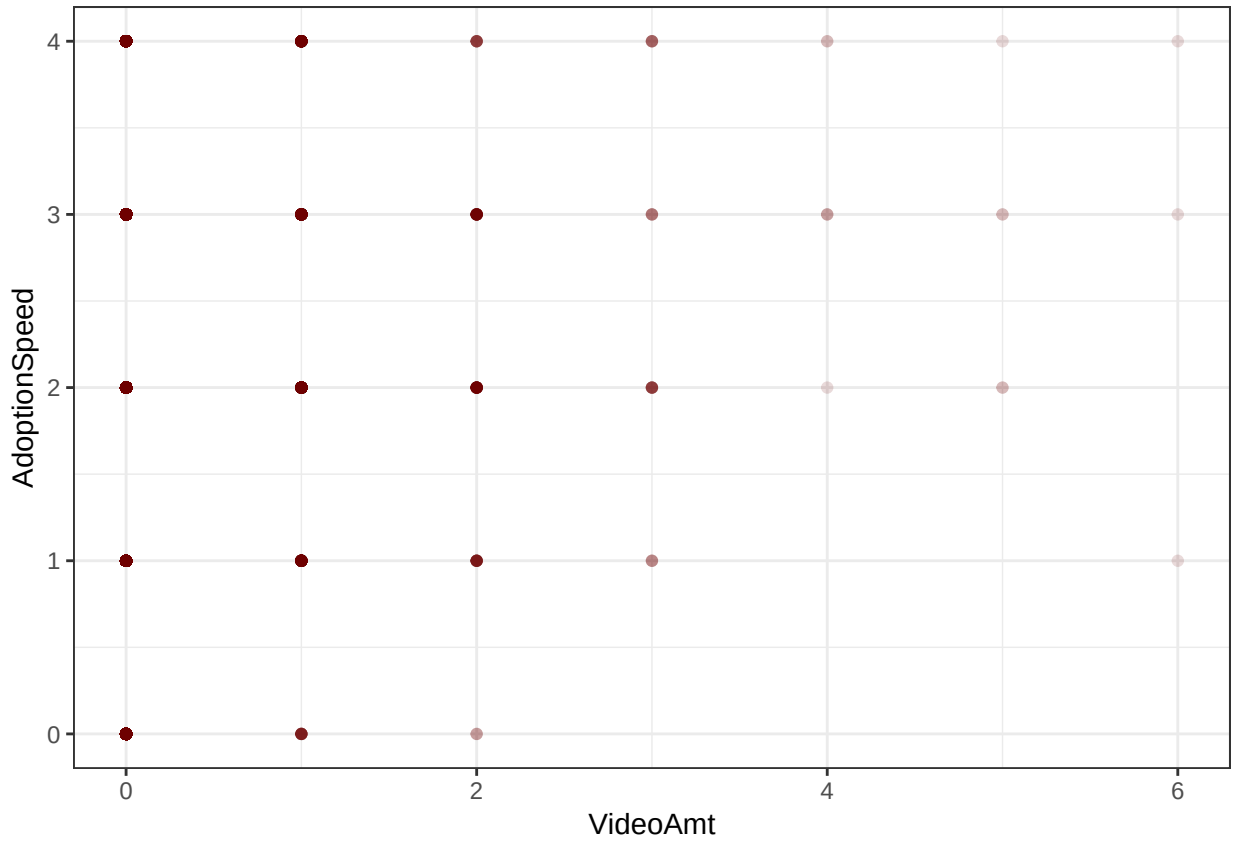
- `Fee` vs. `AdoptionSpeed`: by reading the plot, even though most adoptions of animals are free, we still can see such data pattern that most animals that were adopted immediately on the same day they listed have very low adoption fee.

```
p <- ggplot(df1, aes(x = Fee, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



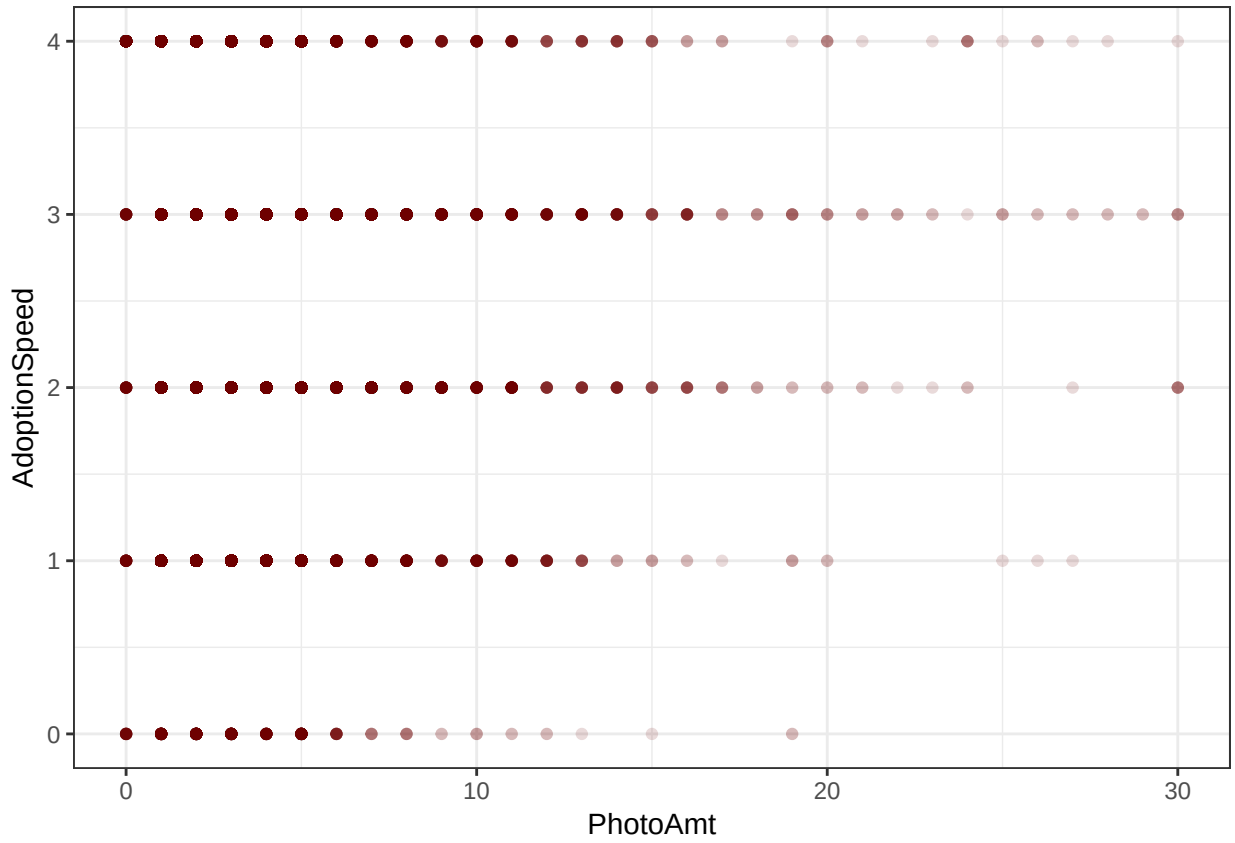
- `VideoAmt` vs. `AdoptionSpeed`: such plot actually might reflect that the later the animal is adopted, the more amount of videos for promoting it.

```
p <- ggplot(df1, aes(x = VideoAmt, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



- `PhotoAmt` vs. `AdoptionSpeed`: Here we see the same relationship that the later the animal is adopted, the more photo amount uploaded for encouraging the adoption of it.

```
p <- ggplot(df1, aes(x = PhotoAmt, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```

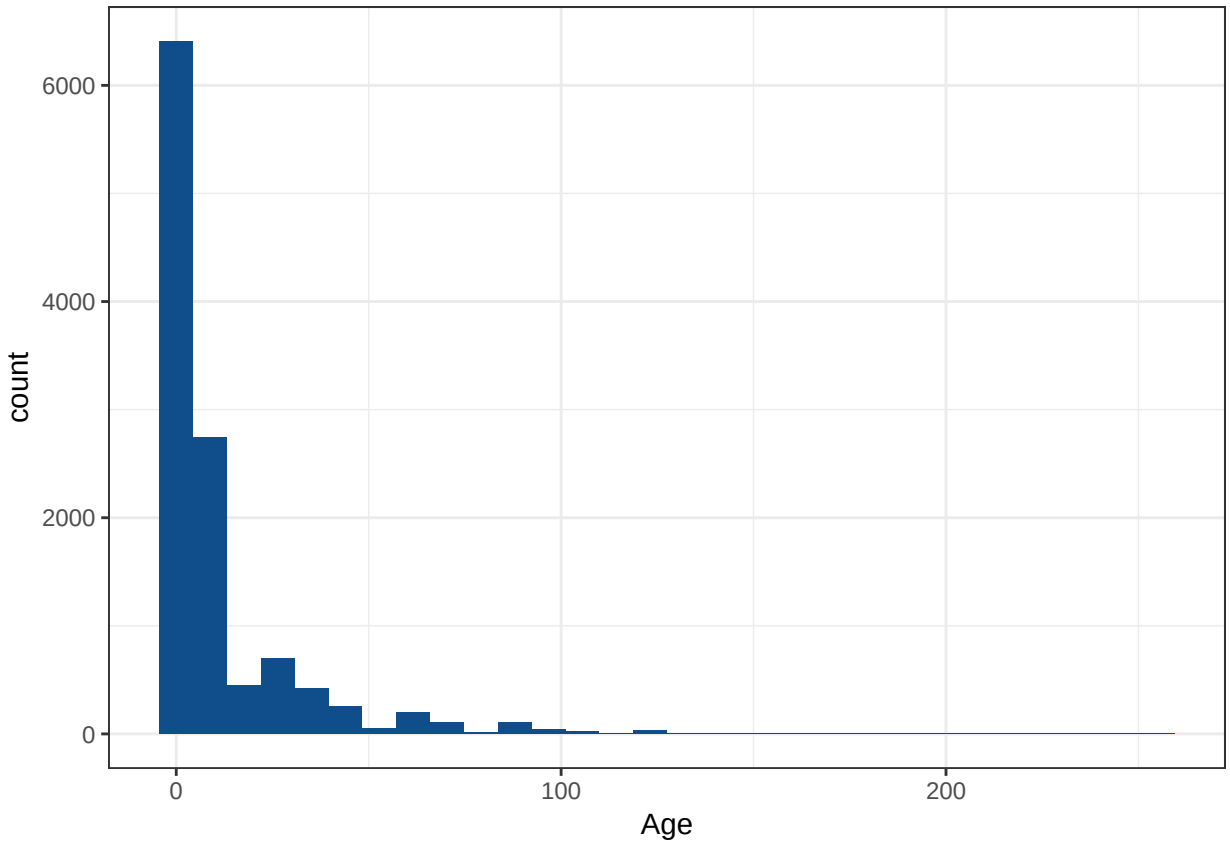


continous variable

- `Age`: Most animals abandoned are very young or even baby.

```
ggplot(df1, aes(x = Age)) +
        geom_histogram(fill = "dodgerblue4")
```

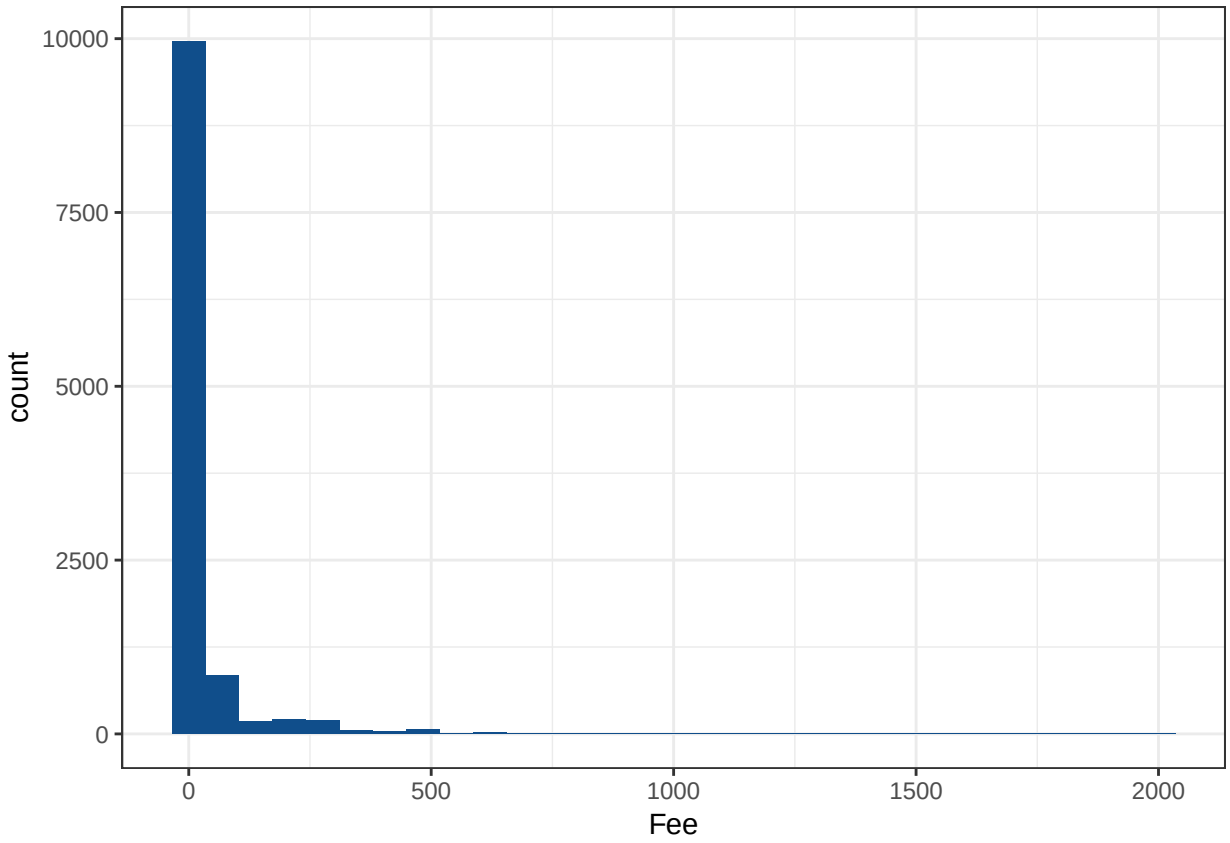
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- `Fee`: Most adoptions are free and even the charged adoptions are still very cheap.

```
ggplot(df1, aes(x = Fee)) +
        geom_histogram(fill = "dodgerblue4")
```

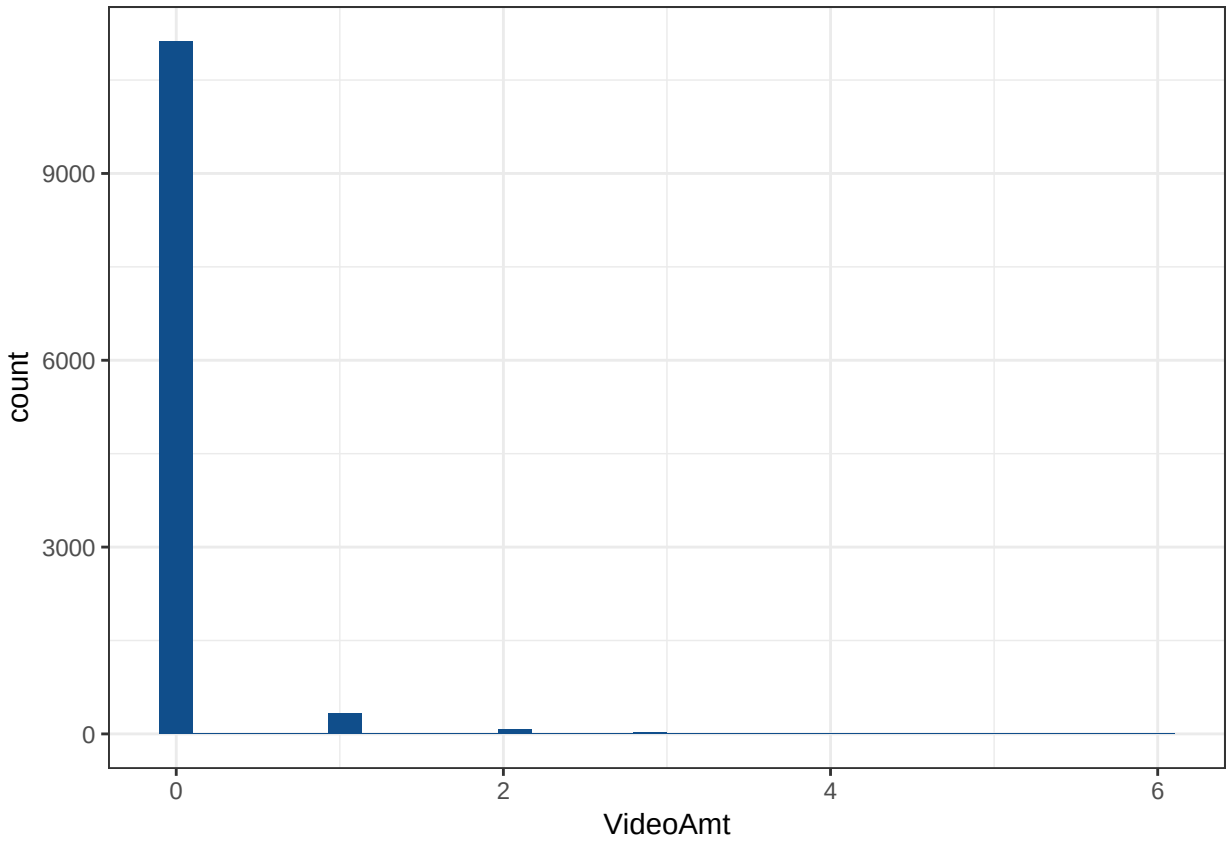
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- `VideoAmt`: Most profiles of animals don't have videos as vivid introductions.

```
ggplot(df1, aes(x = VideoAmt)) +
        geom_histogram(fill = "dodgerblue4")
```

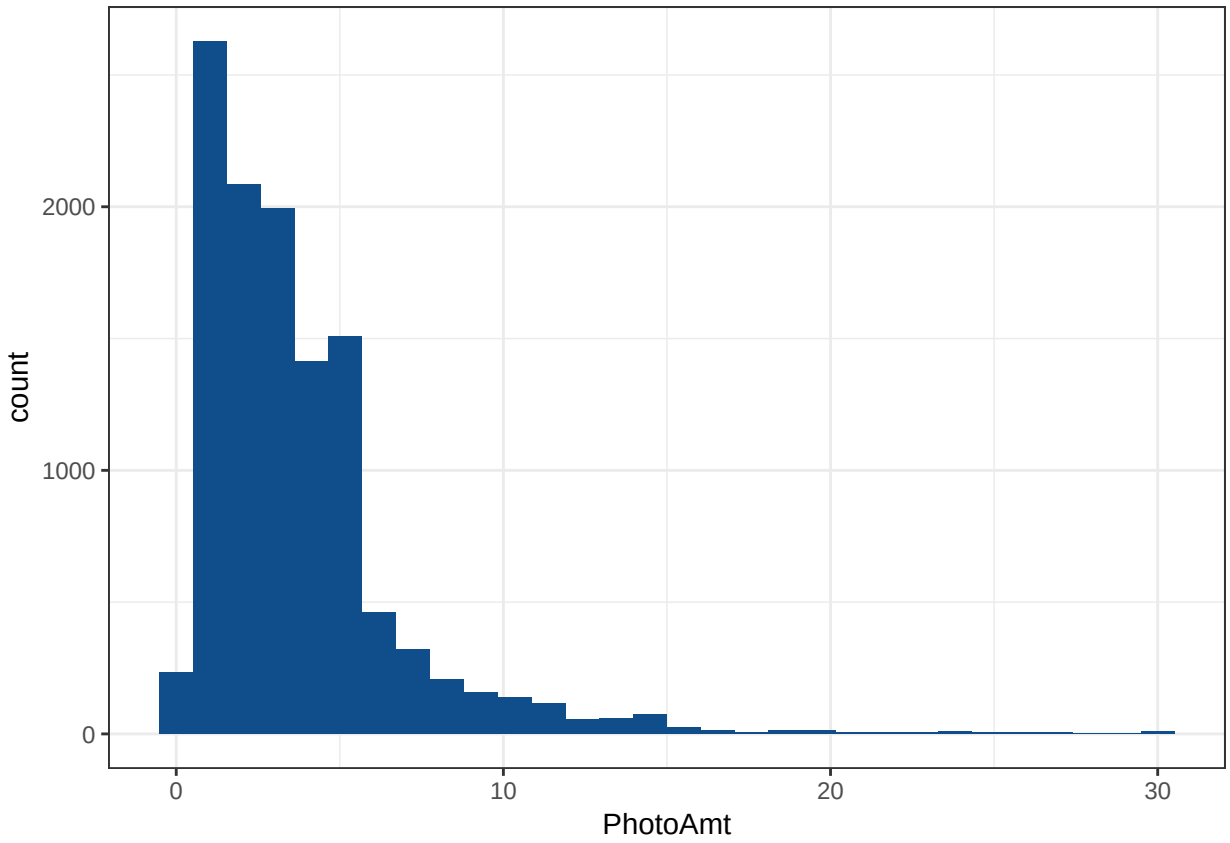
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- `PhotoAmt`: Most profiles have at least one picture of the animal.

```
ggplot(df1, aes(x = PhotoAmt)) +
        geom_histogram(fill = "dodgerblue4")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

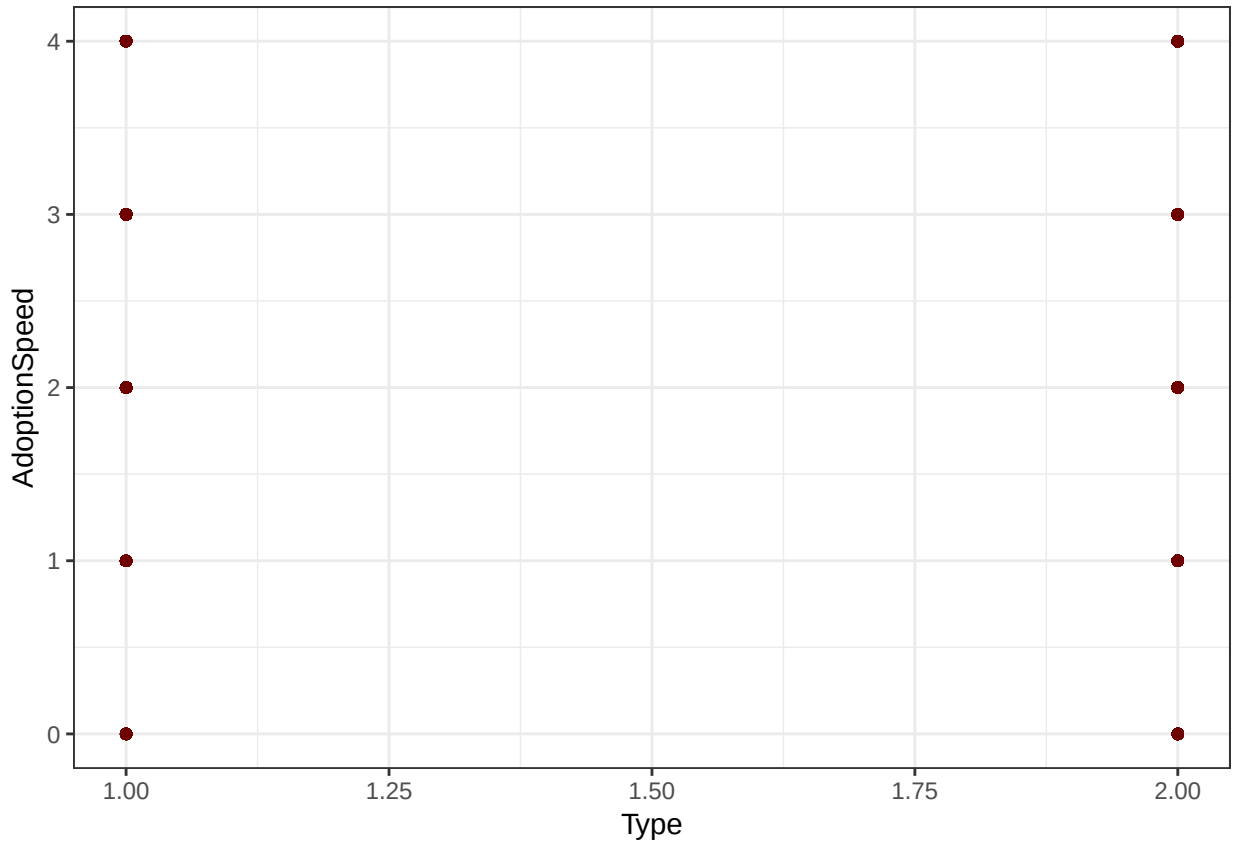


categorical variables vs. target

Let's plot the relationships between categorical variables and target variable, `AdoptionSpeed`.

- `Type`(1 = Dog, 2 = Cat) vs. `AdoptionSpeed`: There are no difference between the adoption speed of dogs and cats.

```
p <- ggplot(df1, aes(x = Type, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



- `Breed1` vs. `AdoptionSpeed`: Since the `Breed1` less than 240 and equals to 307 are dogs, others are cats. We can see that, except `Breed1` equals to 307, dogs generally have a slightly slower adoption speed than cats.

```
Breed <- count(df1$Breed1)
Breed <- Breed[order(-Breed$freq),]
Breed
```

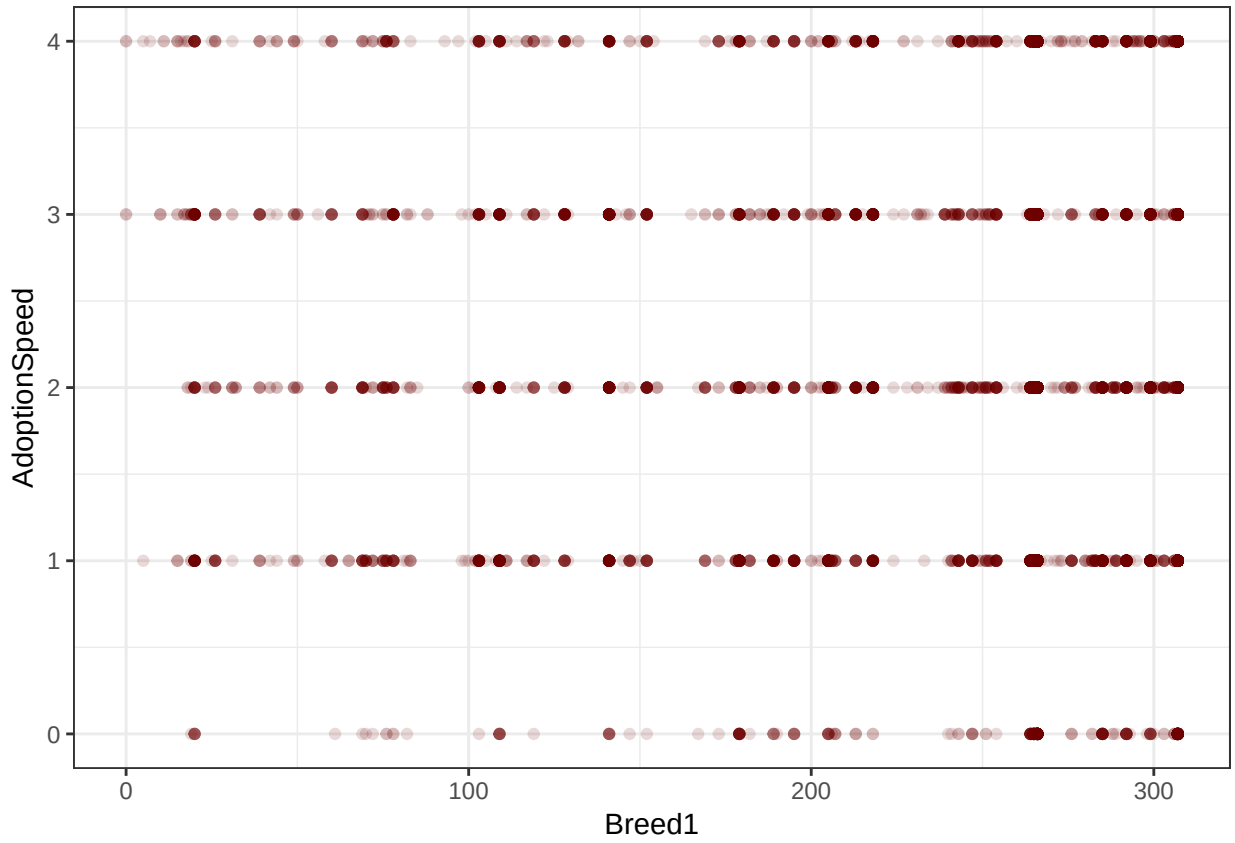
```
##       x freq
## 166 307 4630
## 130 266 2607
## 129 265  865
## 158 299  244
## 128 264  223
## 151 292  203
## 90  205  178
## 147 285  178
## 60  141  175
## 76  179  161
## 98  218  143
## 49  109  134
## 46  103   94
## 11   20   86
## 81  189   81
## 94  213   77
## 122 254   69
## 84  195   66
```

```
## 111 243   66
## 57  128   61
## 67  152   59
## 145 283   58
## 34   78   54
## 115 247   53
## 165 306   45
## 28   69   38
## 53  119   35
## 162 303   35
## 33   76   32
## 25   60   30
## 139 276   29
## 15   26   25
## 119 251   25
## 32   75   24
## 92  207   24
## 91  206   22
## 18   39   21
## 75  178   21
## 150 289   21
## 77  182   20
## 109 241   20
## 120 252   19
## 64  147   17
## 72  169   17
## 73  173   16
## 21   49   14
## 31   72   14
## 110 242   13
## 117 249   13
## 149 288   13
## 159 300   13
## 29   70   12
## 86  200   12
## 107 239   12
## 144 282   12
## 118 250   11
## 22   50   10
## 52  117   10
## 10   19    9
## 37   83    9
## 6    15    8
## 16   31    8
## 20   44    7
## 154 295    7
## 9    18    6
## 50  111    6
## 102 231    6
## 140 277    6
## 19   42    5
## 36   82    5
## 45  102    5
## 78  185    5
```

```
## 82  190    5
## 87  202    5
## 108 240    5
## 112 244    5
## 136 272    5
## 137 273    5
## 138 274    5
## 155 296    5
## 163 304    5
## 164 305    5
## 1     0    4
## 8    17    4
## 44  100    4
## 58  129    4
## 121 253    4
## 142 280    4
## 146 284    4
## 153 294    4
## 160 301    4
## 4    10    3
## 17   32    3
## 27   65    3
## 47  105    3
## 54  122    3
## 69  155    3
## 79  187    3
## 88  203    3
## 89  204    3
## 99  224    3
## 100 227    3
## 152 293    3
## 2     5    2
## 5    11    2
## 14   25    2
## 24   58    2
## 30   71    2
## 39   88    2
## 42   98    2
## 48  108    2
## 51  114    2
## 59  132    2
## 62  145    2
## 66  150    2
## 71  167    2
## 96  215    2
## 103 232    2
## 104 233    2
## 105 234    2
## 106 237    2
## 113 245    2
## 114 246    2
## 116 248    2
## 125 260    2
## 131 267    2
```

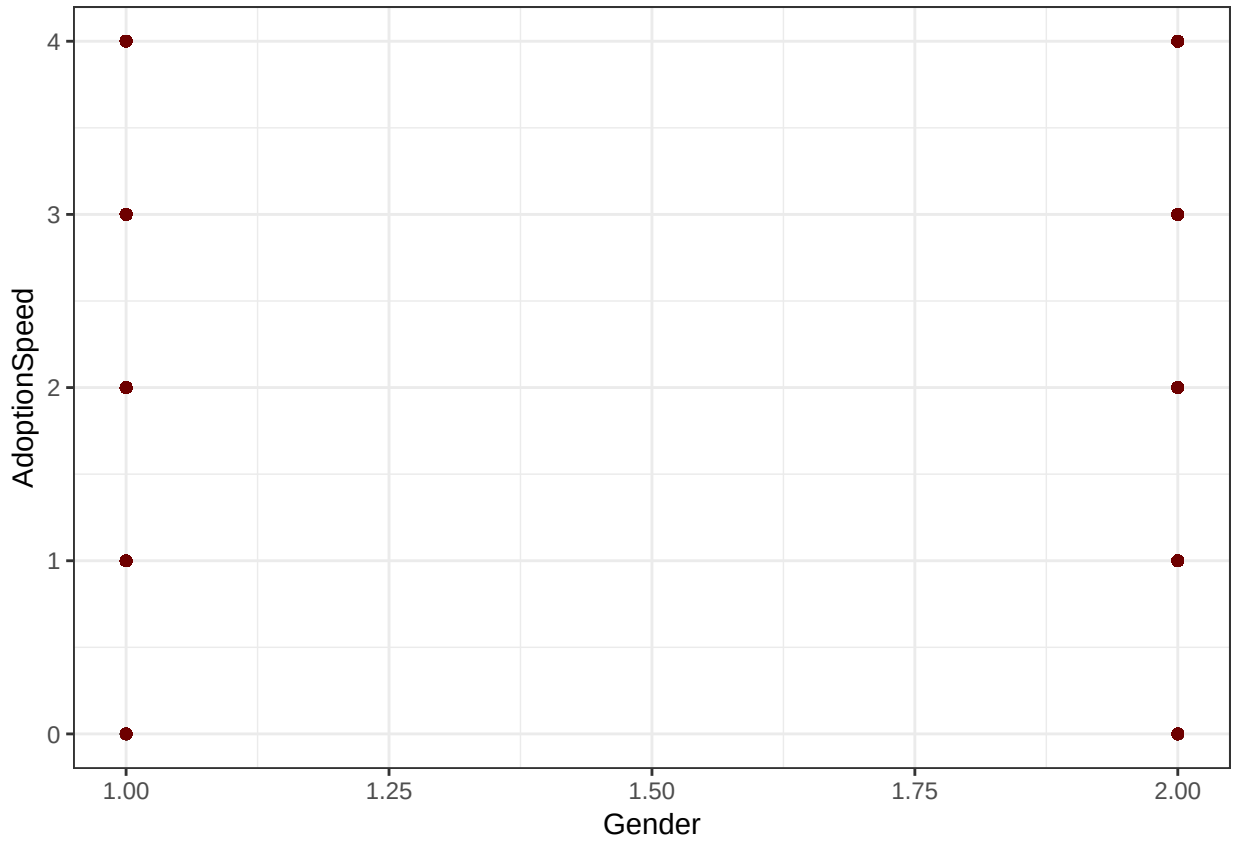
```
## 134 270    2
## 135 271    2
## 141 279    2
## 143 281    2
## 156 297    2
## 3     7    1
## 7    16    1
## 12   23    1
## 13   24    1
## 23   56    1
## 26   61    1
## 35   81    1
## 38   85    1
## 40   93    1
## 41   97    1
## 43   99    1
## 55  123    1
## 56  125    1
## 61  143    1
## 63  146    1
## 65  148    1
## 68  154    1
## 70  165    1
## 74  176    1
## 80  188    1
## 83  192    1
## 85  199    1
## 93  212    1
## 95  214    1
## 97  217    1
## 101 228    1
## 123 256    1
## 124 257    1
## 126 262    1
## 127 263    1
## 132 268    1
## 133 269    1
## 148 287    1
## 157 298    1
## 161 302    1
```

```
p <- ggplot(df1, aes(x = Breed1, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



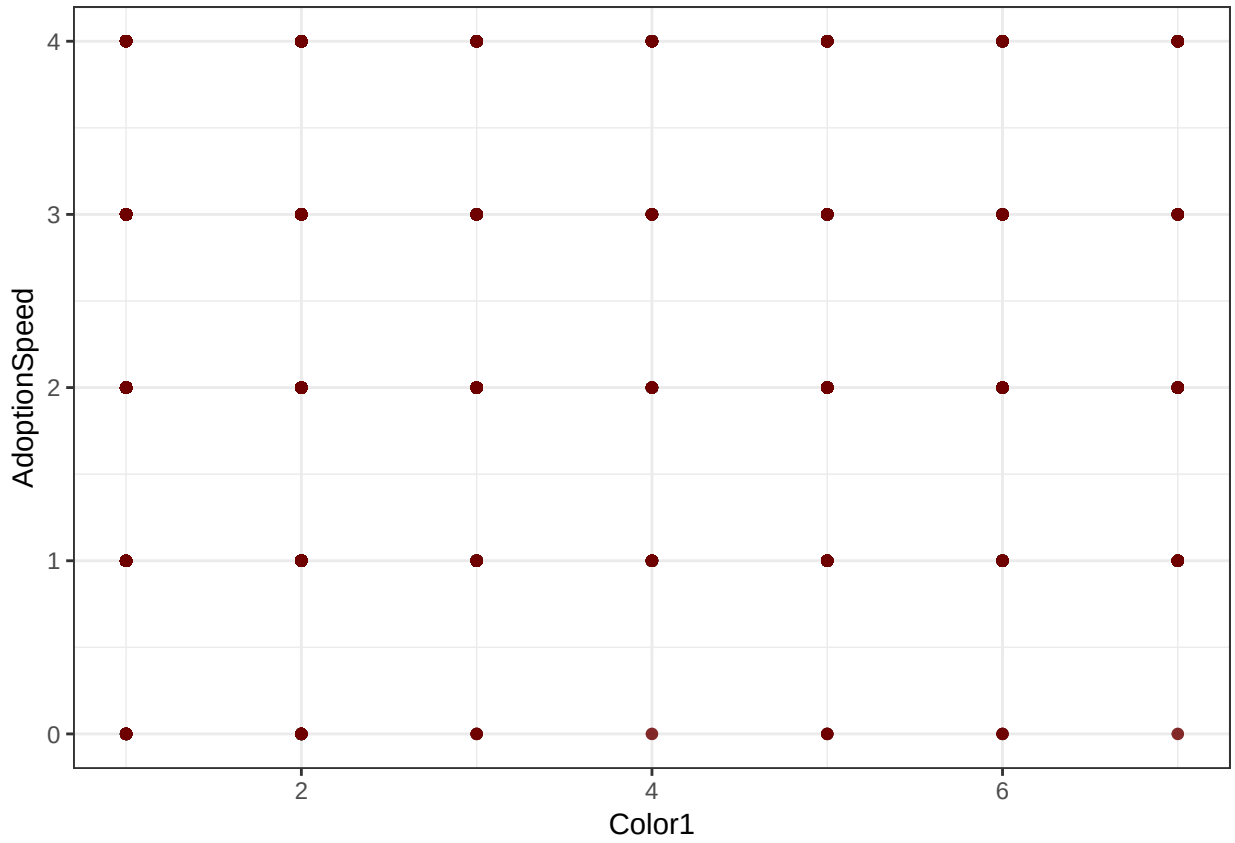
- `Gender`(1 = Male, 2 = Female) vs. `AdoptionSpeed`: There are no difference between the adoption speed of different genders.

```
p <- ggplot(df1, aes(x = Gender, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```

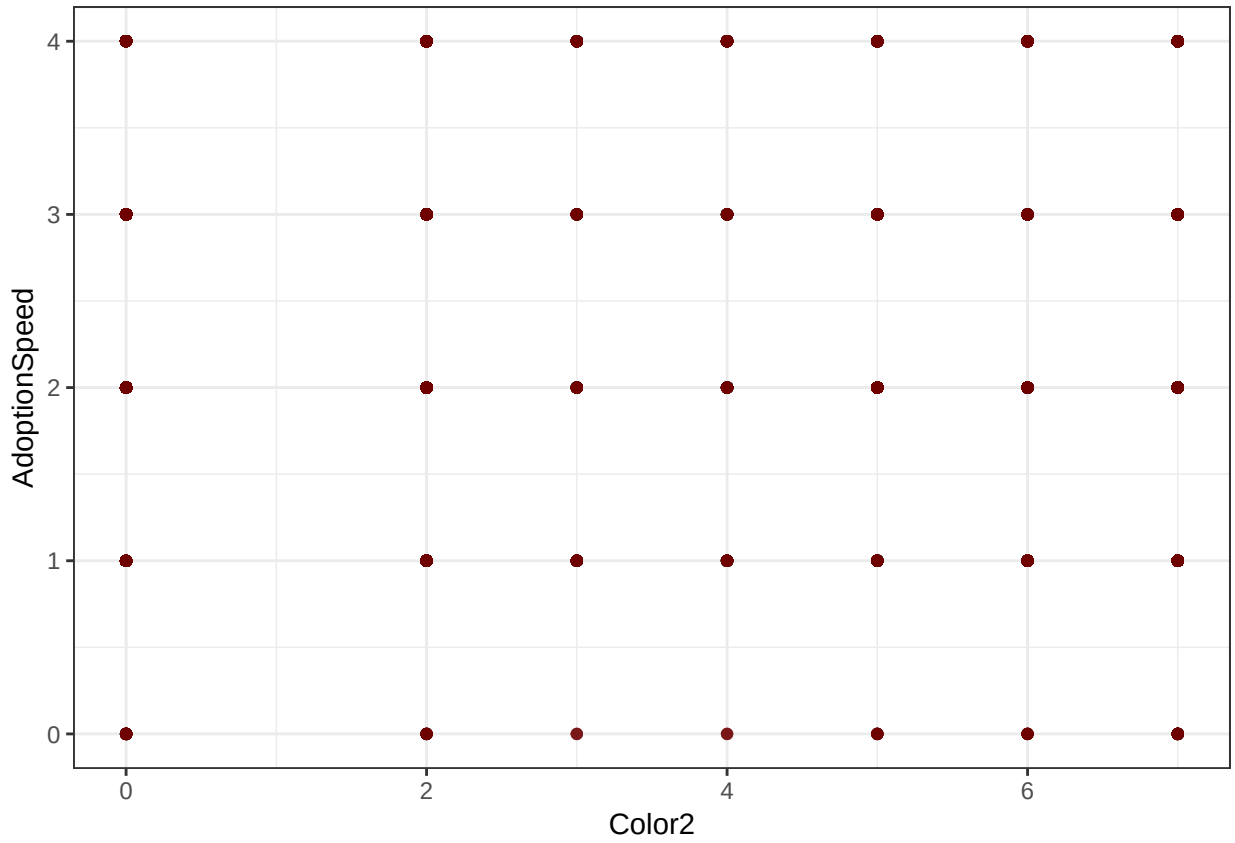


- `Color1`&`Color2`&`Color3` vs. `AdoptionSpeed`: There are no difference among the adoption speed of different colors of animals.

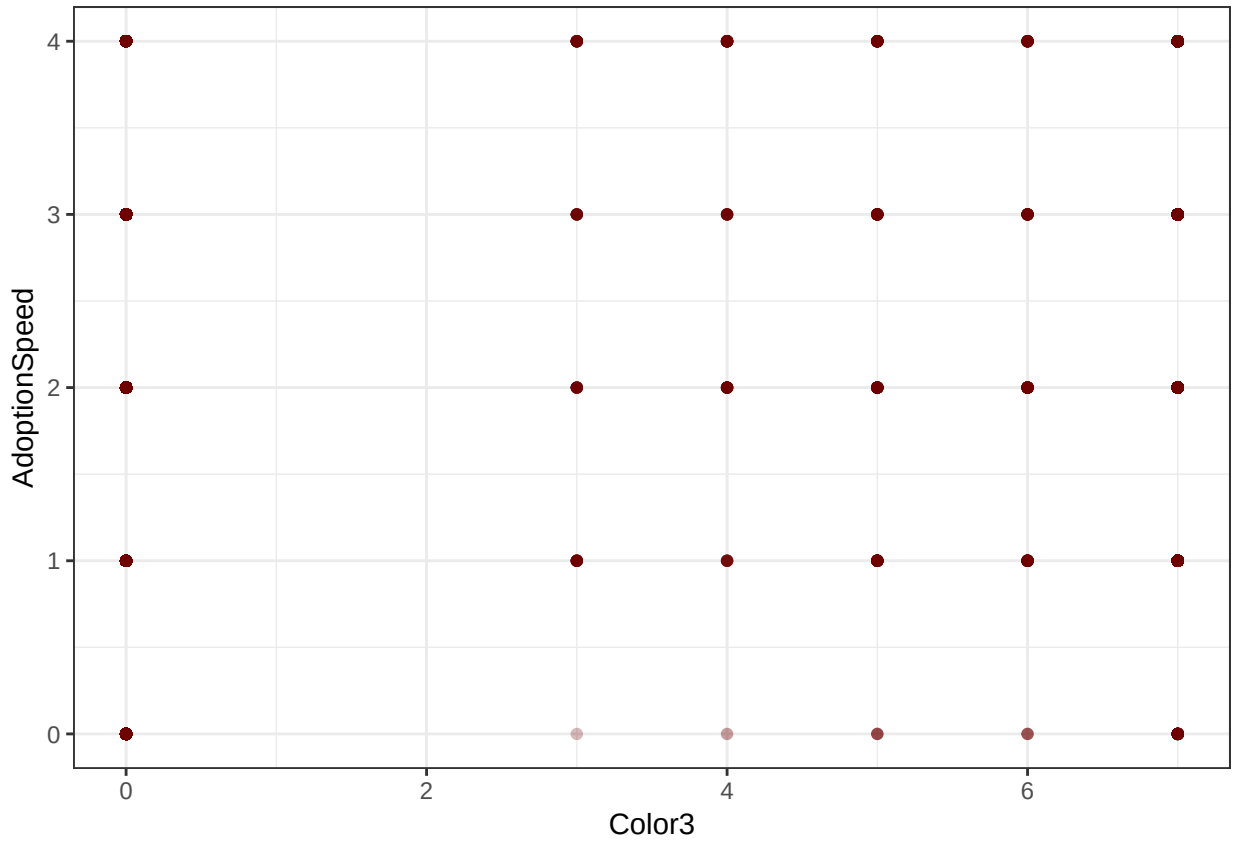
```
p1 <- ggplot(df1, aes(x = Color1, y = AdoptionSpeed))
p1 + geom_point(alpha = .15, col = "#6e0000")
```




```
p2 <- ggplot(df1, aes(x = Color2, y = AdoptionSpeed))
p2 + geom_point(alpha = .15, col = "#6e0000")
```

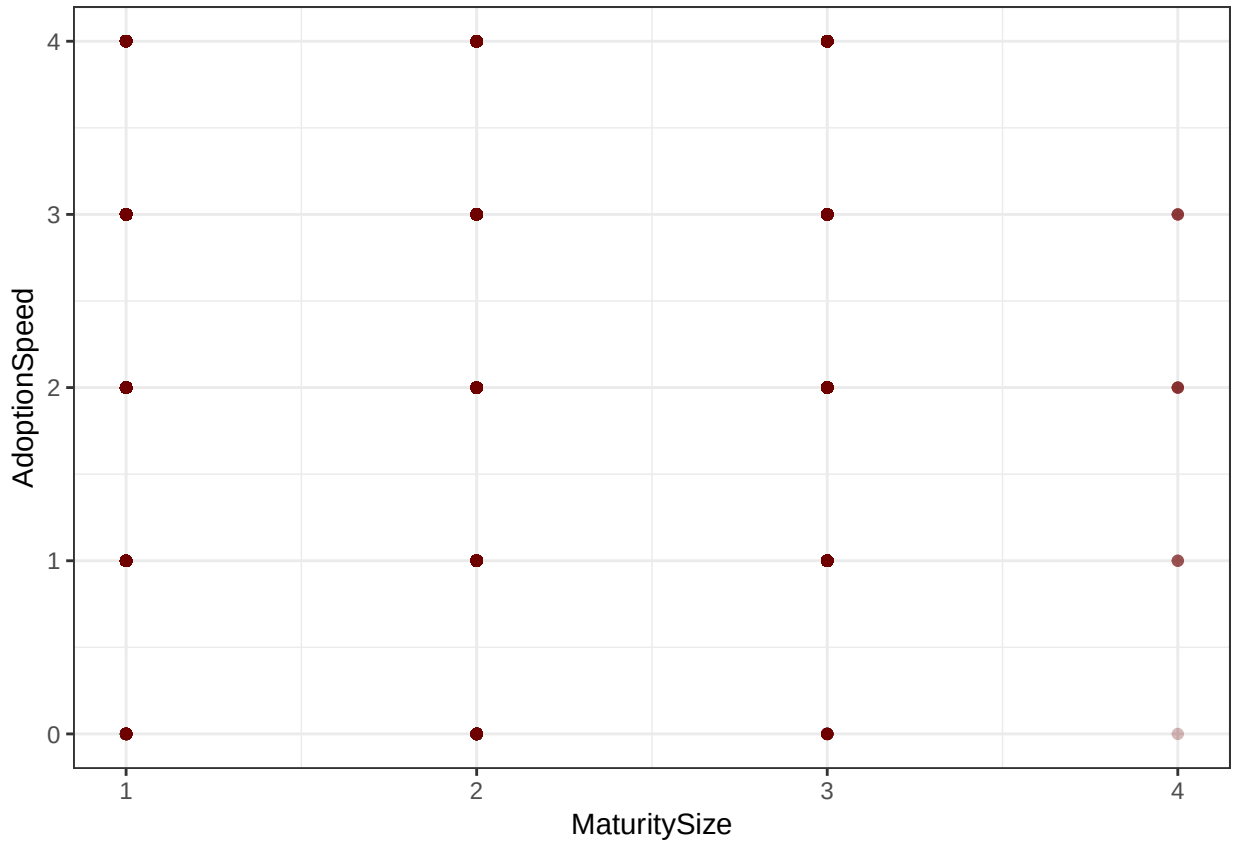



```
p3 <- ggplot(df1, aes(x = Color3, y = AdoptionSpeed))
p3 + geom_point(alpha = .15, col = "#6e0000")
```



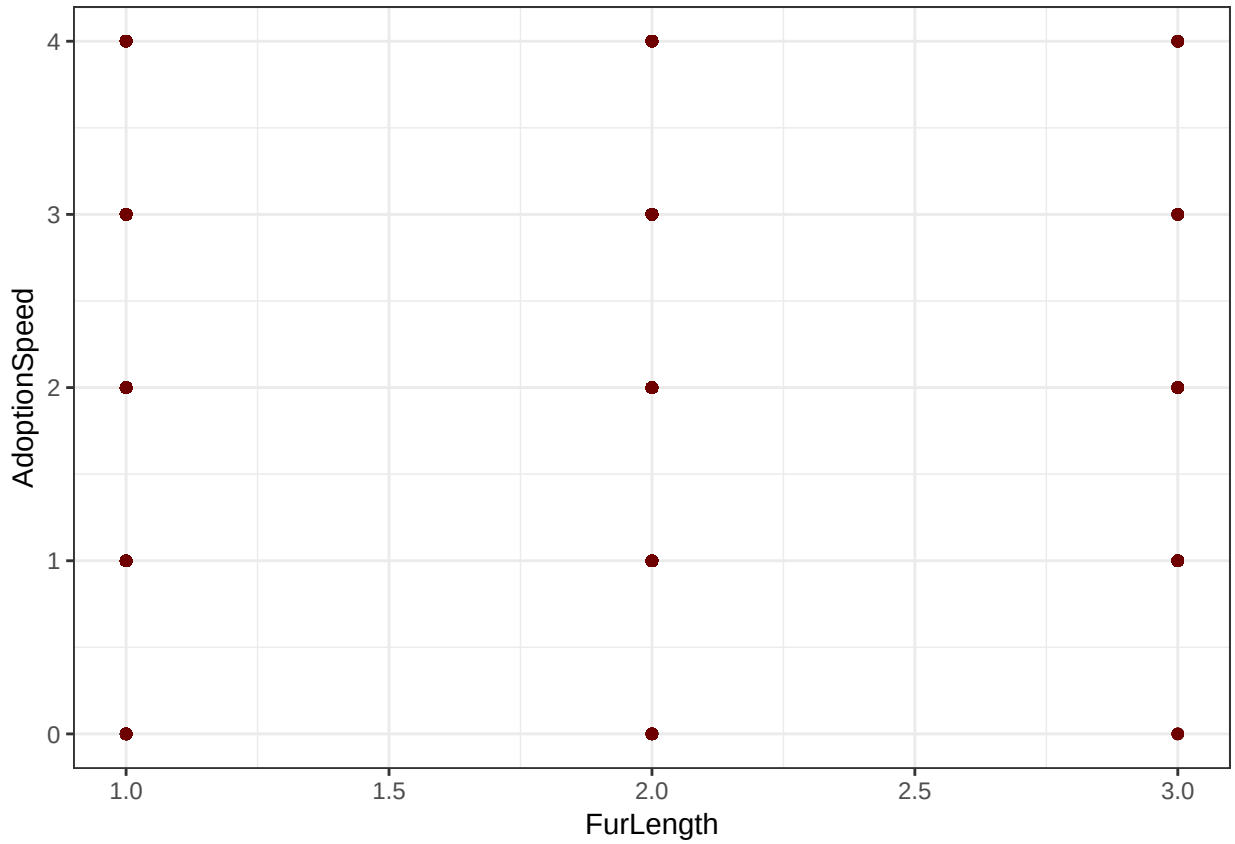
- `MaturitySize` (1 = Small, 2 = Medium, 3 = Large, 4 = Extra Large, 0 = Not Specified) vs. `AdoptionSpeed`: Animals with extra large maturity size usually have to wait 1 week to 3 months for being adopted.

```
p <- ggplot(df1, aes(x = MaturitySize, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



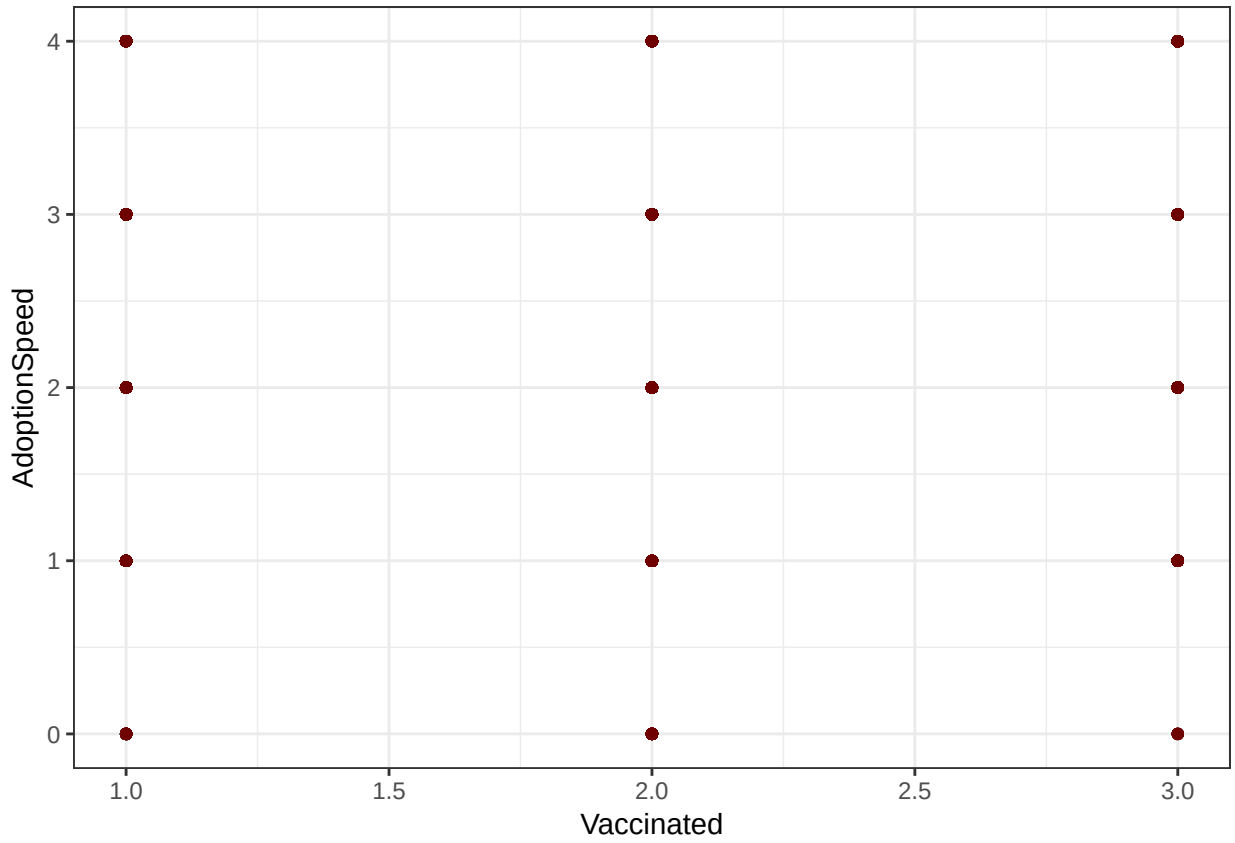
- `FurLength`(1 = Short, 2 = Medium, 3 = Long, 0 = Not Specified) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different fur lengths of animals.

```
p <- ggplot(df1, aes(x = FurLength, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



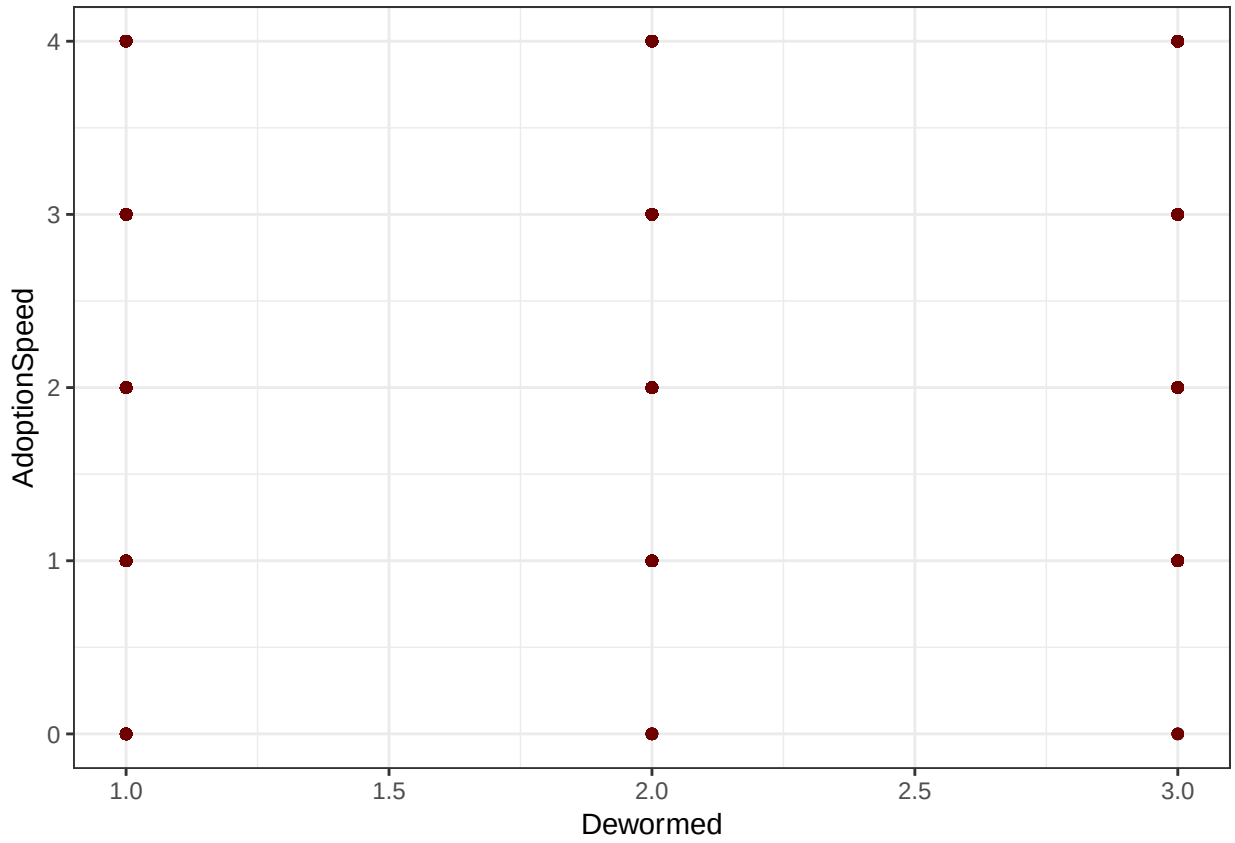
- `Vaccinated`(1 = Yes, 2 = No, 3 = Not Sure) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different vaccinated conditions of animals.

```
p <- ggplot(df1, aes(x = Vaccinated, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



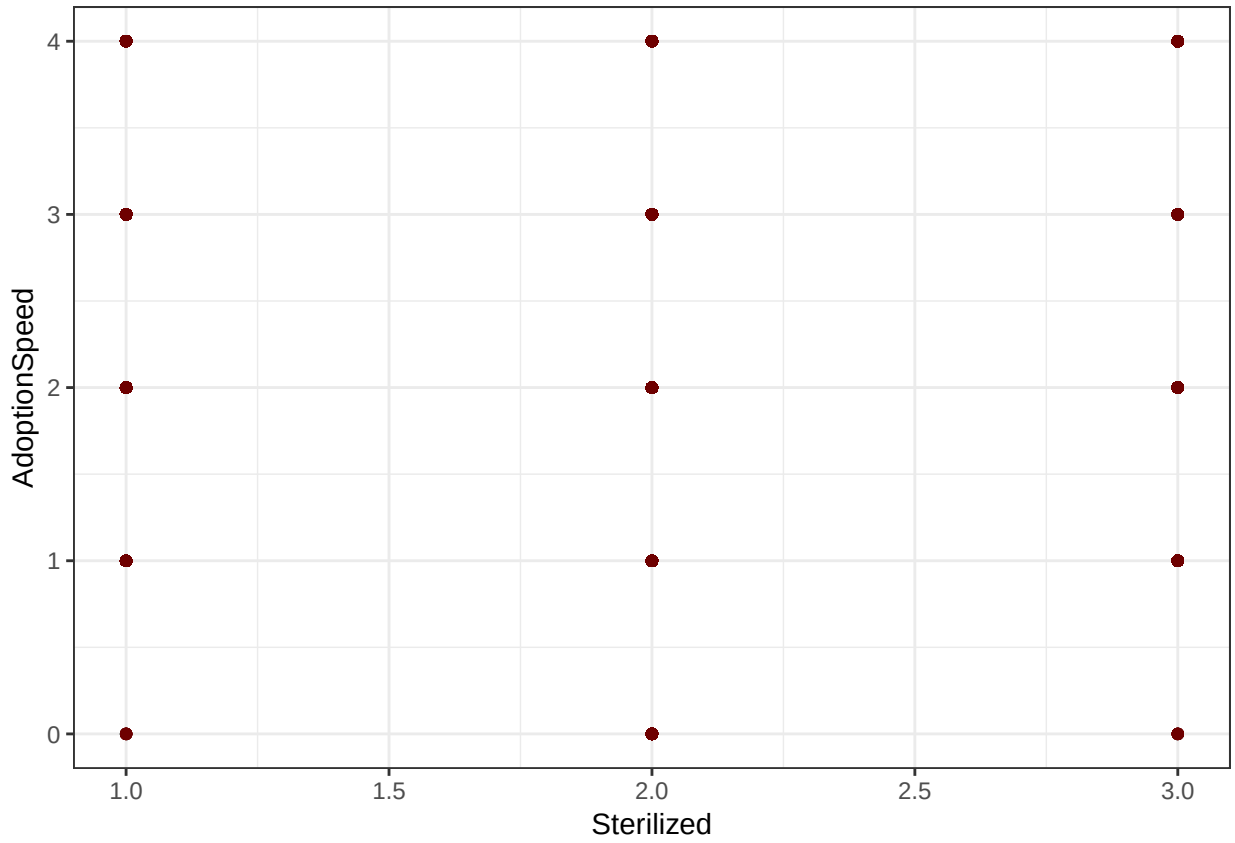
- `Dewormed`(1 = Yes, 2 = No, 3 = Not Sure) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different dewormed conditions of animals.

```
p <- ggplot(df1, aes(x = Dewormed, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



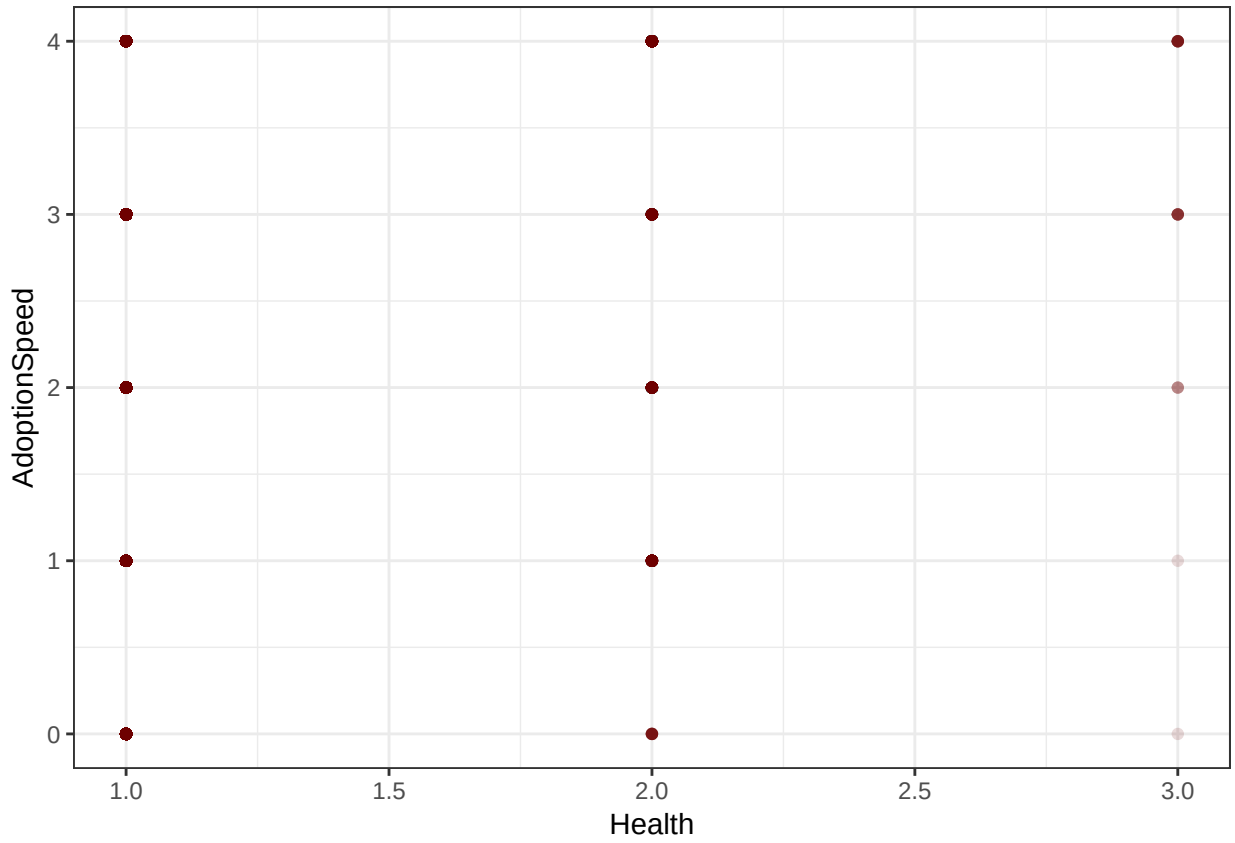
- `Sterilized`(1 = Yes, 2 = No, 3 = Not Sure) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different sterilized conditions of animals.

```
p <- ggplot(df1, aes(x = Sterilized, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



- `Health`(1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified) vs. `AdoptionSpeed`: It's obvious to see that seriously injured animals usually have to wait more than at least 1 month for being adopted.

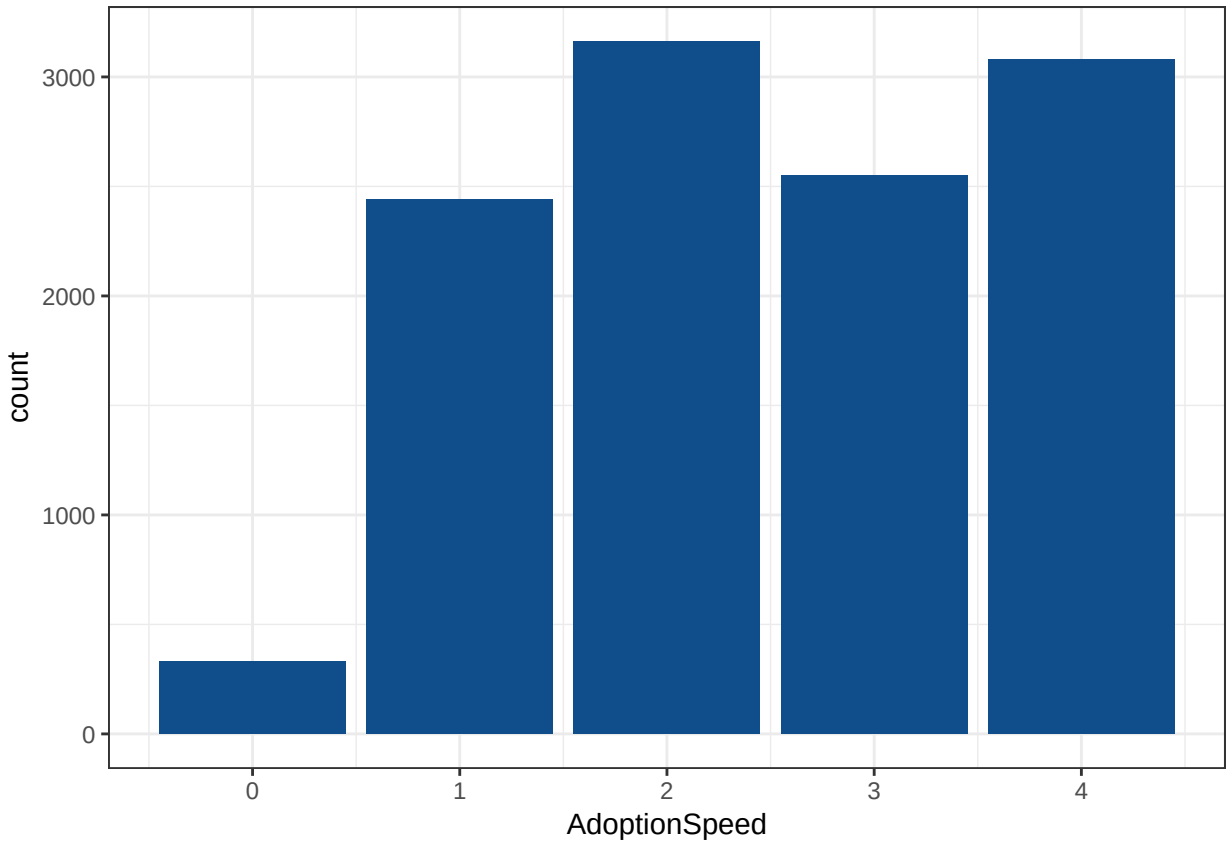
```
p <- ggplot(df1, aes(x = Health, y = AdoptionSpeed))

p + geom_point(alpha = .15, col = "#6e0000")
```



categorical variables

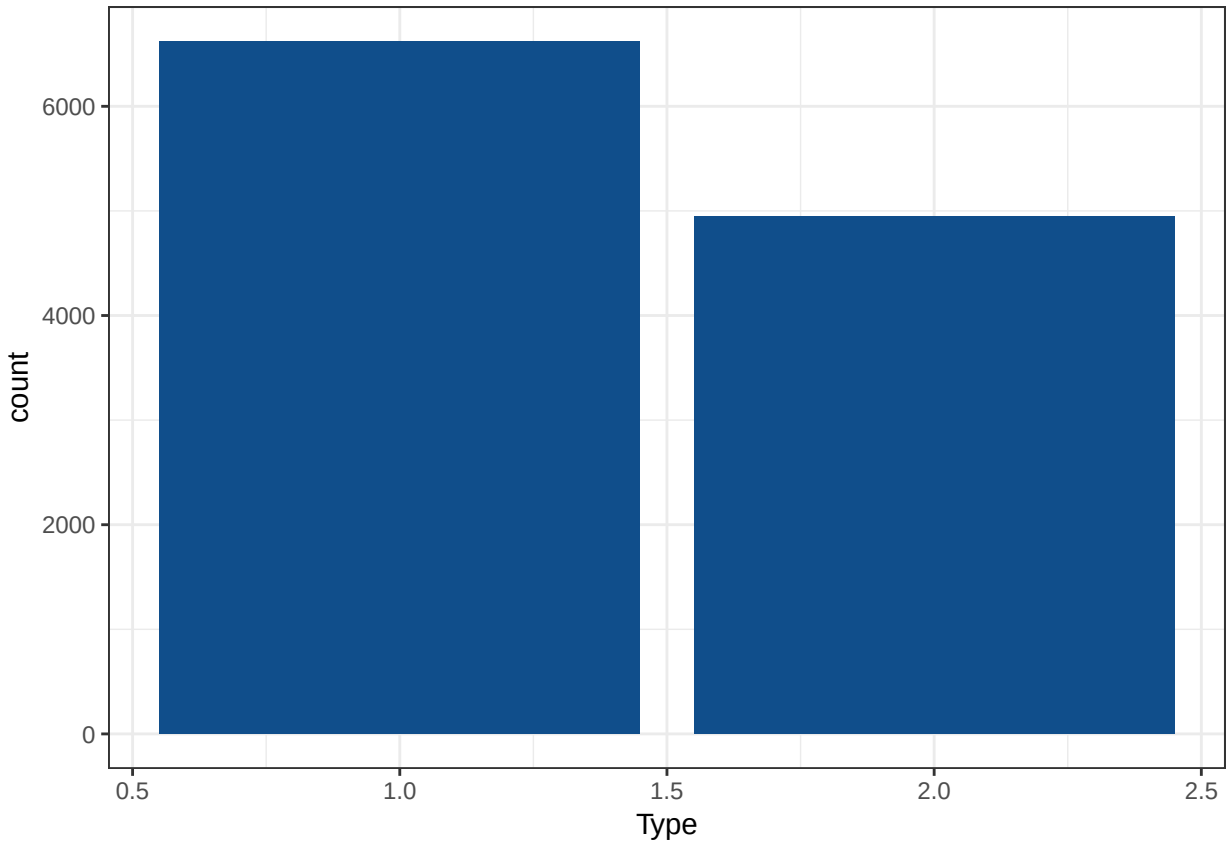
Since the target is also categorical, here is the data pattern of `AdoptionSpeed`: We can know that most adoptions happened after 1 month of the animals 'profiles created. And there still have lots of animals not adopted after even 100 days. - The values of `AdoptionSpeed`: 0 - Pet was adopted on the same day as it was listed. 1 - Pet was adopted between 1 and 7 days (1st week) after being listed. 2 - Pet was adopted between 8 and 30 days (1st month) after being listed. 3 - Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed. 4 - No adoption after 100 days of being listed. (There are no pets in this dataset that waited between 90 and 100 days).

```
p <- ggplot(df1, aes(x = AdoptionSpeed)) + geom_bar(fill = "dodgerblue4")
p
```



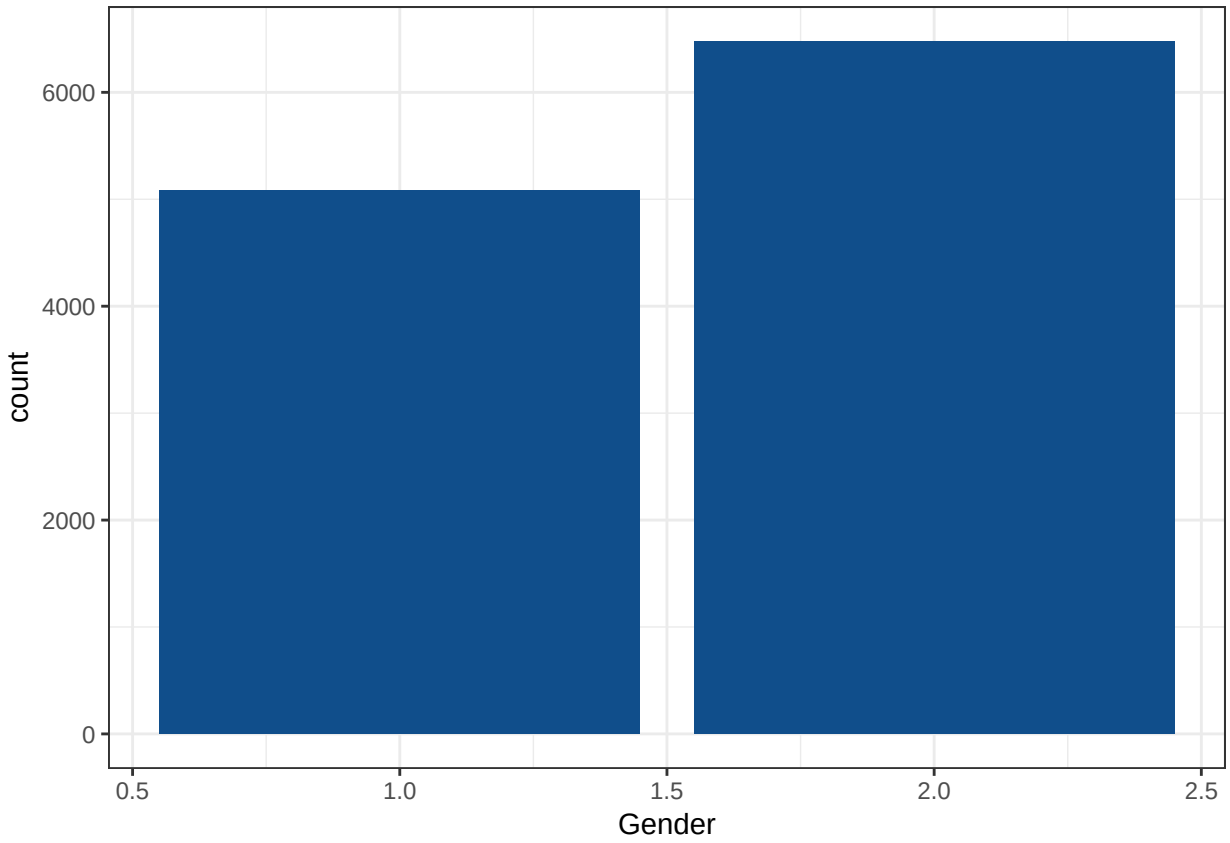
- `Type`(1 = Dog, 2 = Cat): There are more dogs than cats abandoned.

```
p <- ggplot(df1, aes(x = Type)) + geom_bar(fill = "dodgerblue4")
p
```



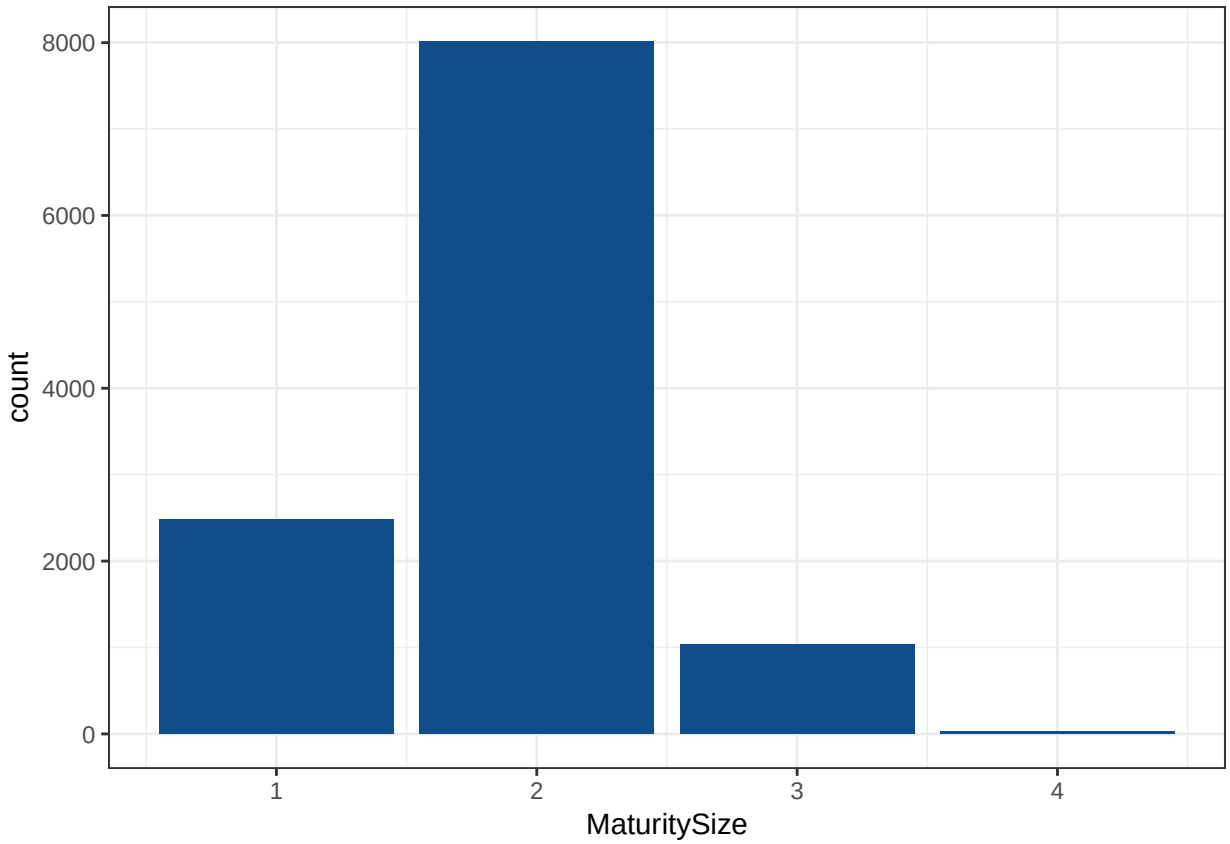
- `Gender`(1 = Male, 2 = Female): There are more female animals than males.

```
p <- ggplot(df1, aes(x = Gender)) + geom_bar(fill = "dodgerblue4")
p
```



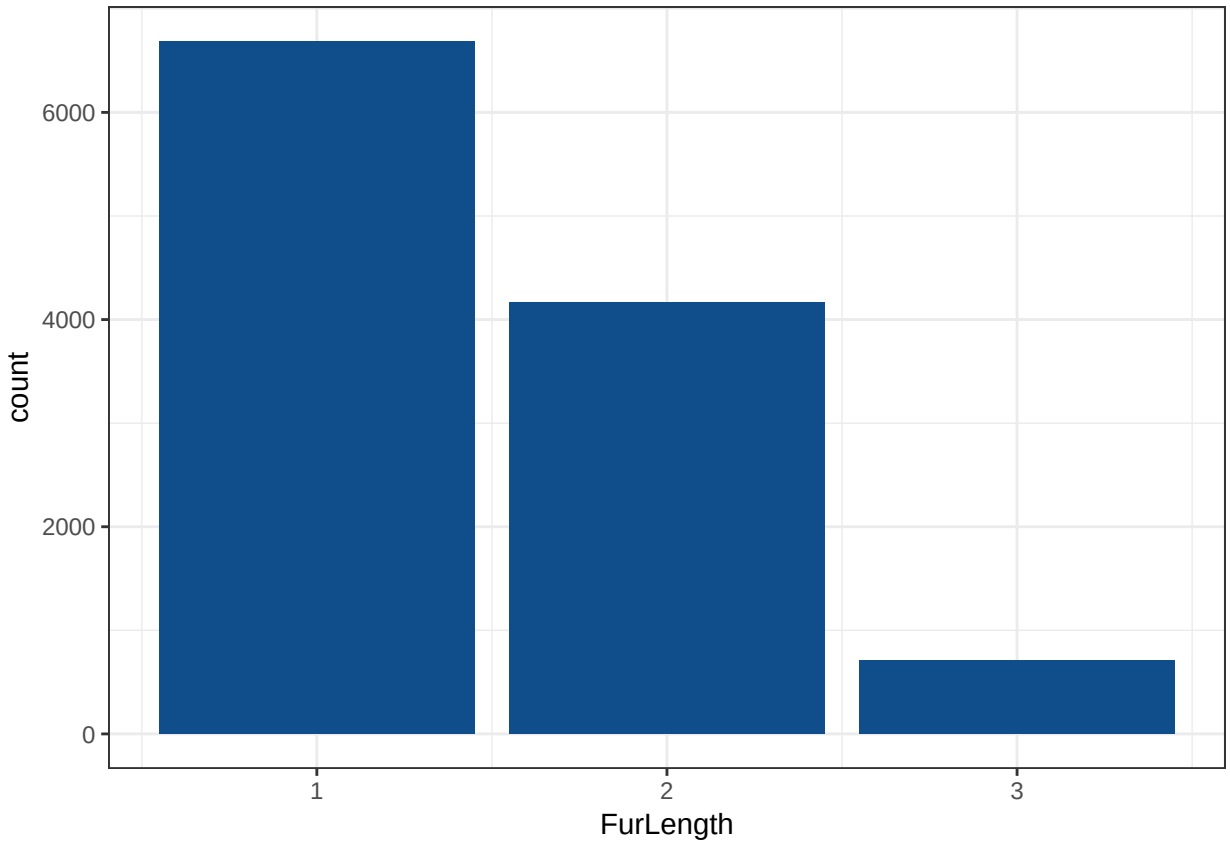
- `MaturitySize`(1 = Small, 2 = Medium, 3 = Large, 4 = Extra Large, 0 = Not Specified): Most animals have medium maturity size.

```
p <- ggplot(df1, aes(x = MaturitySize)) + geom_bar(fill = "dodgerblue4")
p
```



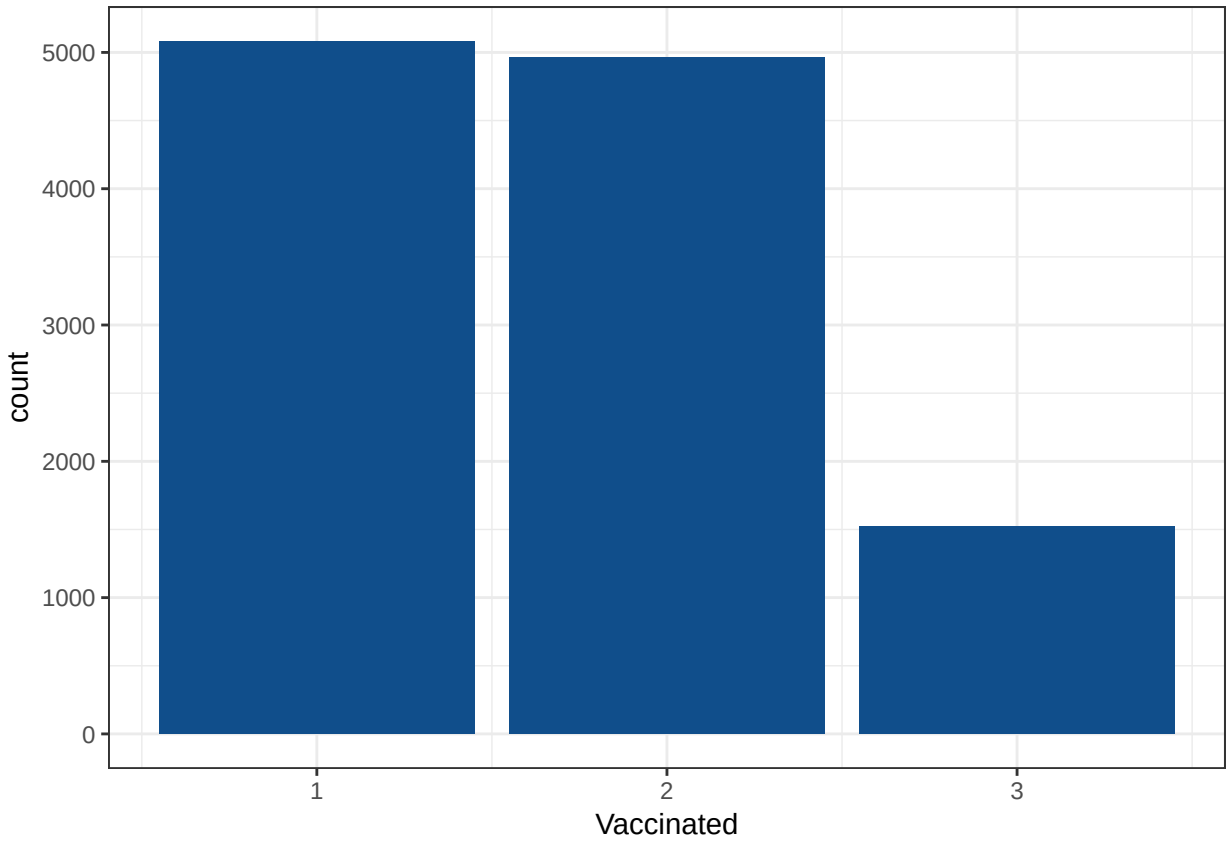
- `FurLength`(1 = Short, 2 = Medium, 3 = Long, 0 = Not Specified): Most animals are short fur.

```
p <- ggplot(df1, aes(x = FurLength)) + geom_bar(fill = "dodgerblue4")
p
```



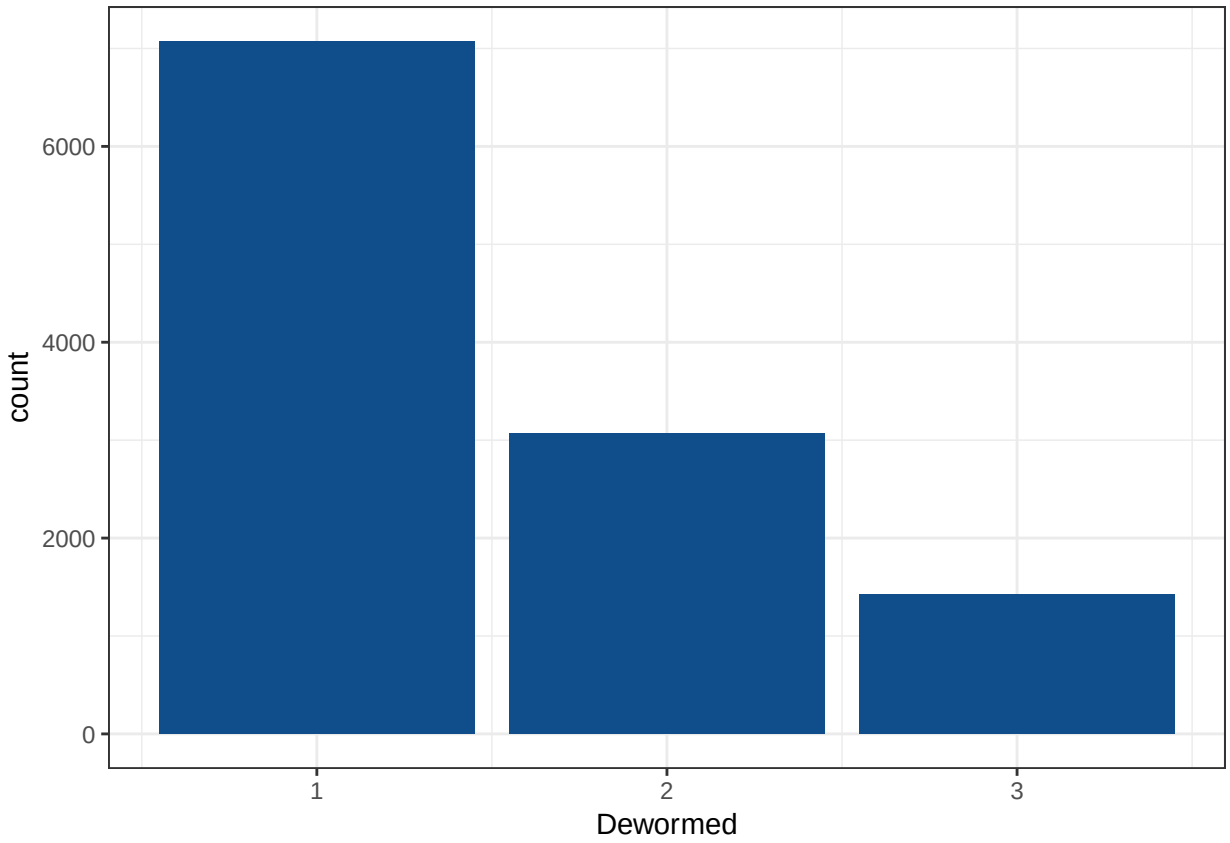
- `Vaccinated`(1 = Yes, 2 = No, 3 = Not Sure): Most animals' vaccinated conditions could be confirmed.

```
p <- ggplot(df1, aes(x = Vaccinated)) + geom_bar(fill = "dodgerblue4")
p
```



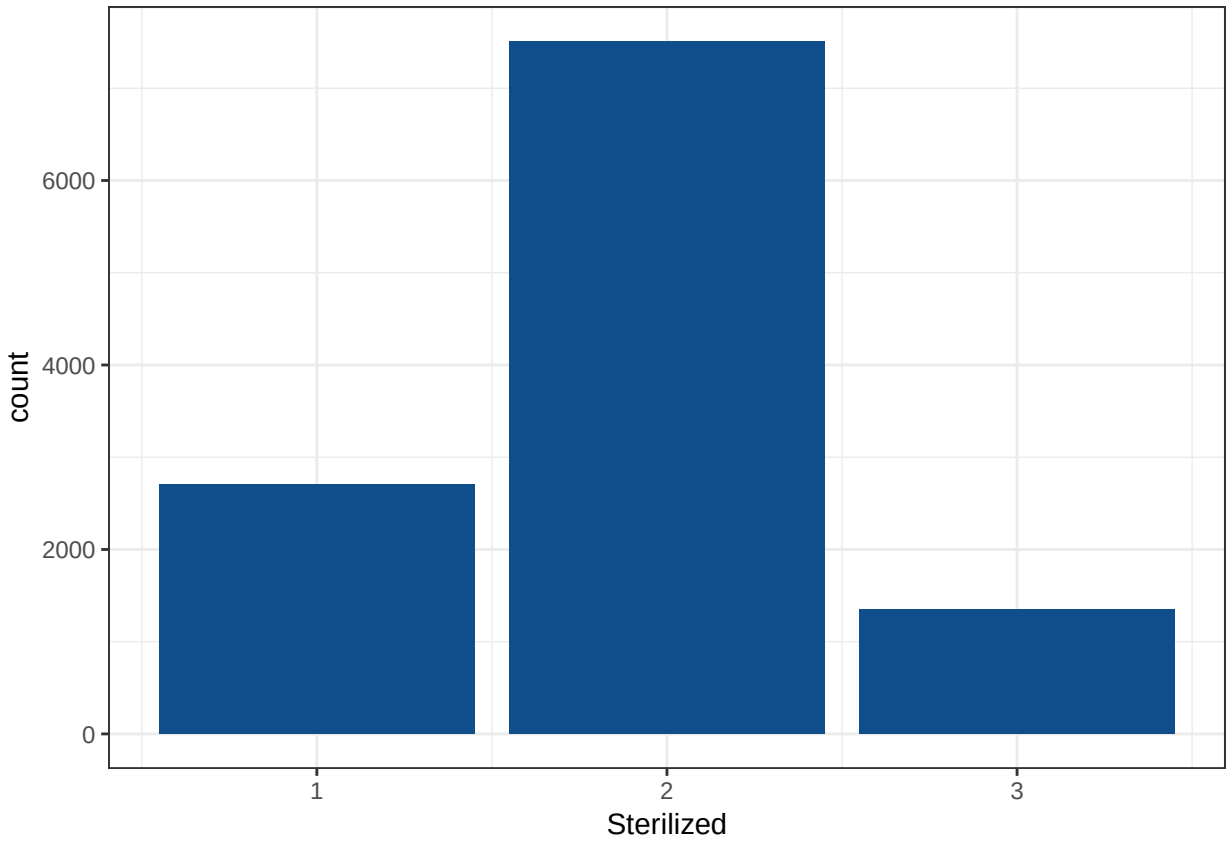
- `Dewormed`(1 = Yes, 2 = No, 3 = Not Sure): Most animals have already been dewormed.

```
p <- ggplot(df1, aes(x = Dewormed)) + geom_bar(fill = "dodgerblue4")
p
```



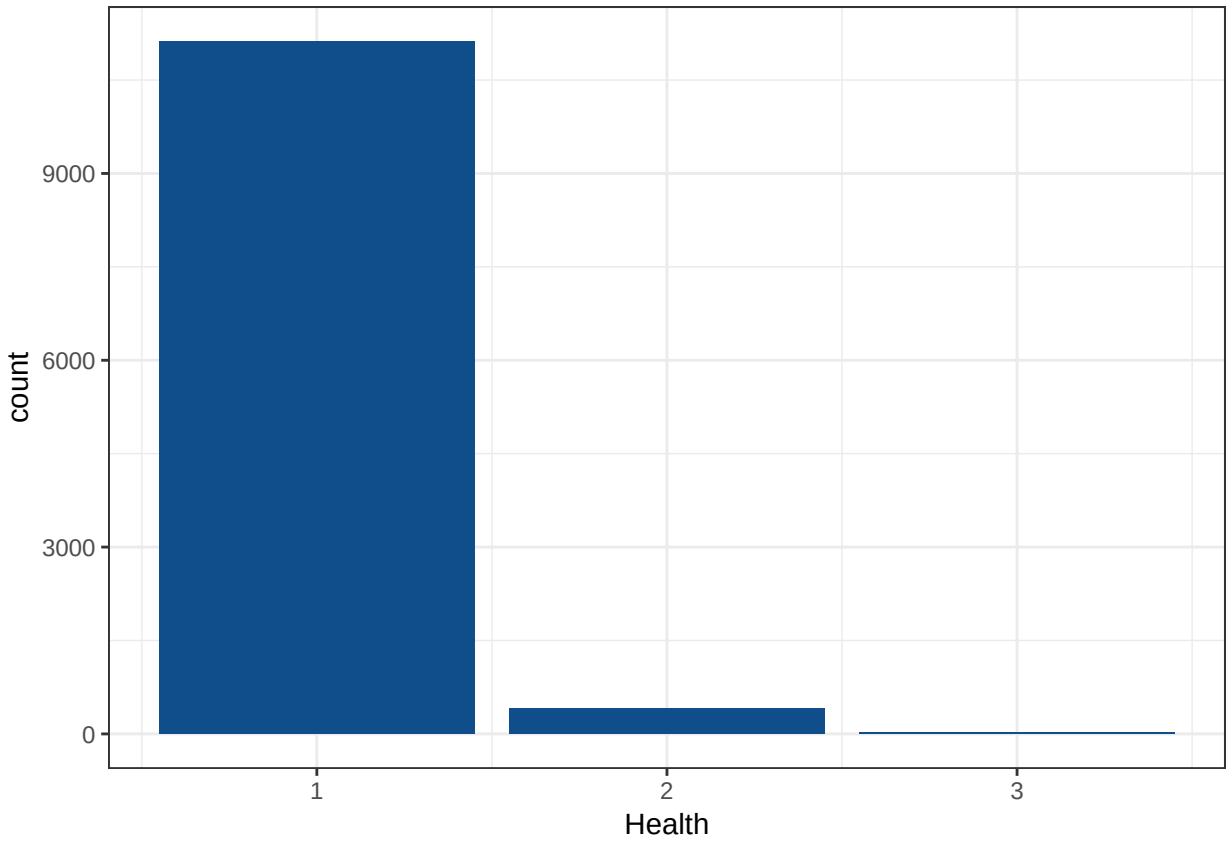
- `Sterilized`(1 = Yes, 2 = No, 3 = Not Sure): Most animals haven't been sterilized yet.

```
p <- ggplot(df1, aes(x = Sterilized)) + geom_bar(fill = "dodgerblue4")
p
```



- `Health`(1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified): Most animals are healthy.

```
p <- ggplot(df1, aes(x = Health)) + geom_bar(fill = "dodgerblue4")
p
```



Data patter: - More dogs than cats; - Much more baby animals; - more female animals; - many black and browm animals; - medium maturity is the most; - short fur length is the most; - most animals' vaccinated situation could be sured; - most animals have already been dewormed; - most animals haven't been sterilized; - most animals are healthy; - most animals' adoption fee is 0; - most animals' profiles don't have video; - most profiles include 1-5 photos; - a few animals were adopted immediatly on the same day of being listed.

Modeling: Supervised Analytics

Logistic Regression

```
adop_multinom <- multinom(AdoptionSpeed ~ .,data = df1)
```

```
## # weights:  95 (72 variable)
## initial  value 18613.149457
## iter  10 value 17055.296975
## iter  20 value 16813.657347
## iter  30 value 16723.892105
## iter  40 value 16662.866250
## iter  50 value 16478.688901
## iter  60 value 16406.395858
## iter  70 value 16330.519470
## iter  80 value 16317.554569
## final  value 16317.553747
## converged
```

```
tidy(adop_multinom)
```

```
## # A tibble: 72 x 6
##    y.level term         estimate std.error statistic  p.value
##    <chr>   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
##  1 1       (Intercept)    30.6    0.190       18.0   9.60e-73
##  2 1       Type            0.737  0.133       -2.30  2.16e- 2
##  3 1       Age             0.991  0.00381     -2.34  1.95e- 2
##  4 1       Breed1          1.00   0.000914     0.252 8.01e- 1
##  5 1       Breed2          0.998  0.000469    -3.93  8.54e- 5
##  6 1       Gender          0.830  0.112       -1.66  9.69e- 2
##  7 1       Color1          0.991  0.0336      -0.259 7.96e- 1
##  8 1       Color2          0.967  0.0216      -1.55  1.22e- 1
##  9 1       Color3          0.961  0.0215      -1.86  6.29e- 2
## 10 1       MaturitySize    1.24   0.0975       2.21  2.73e- 2
## # ... with 62 more rows
```

Let's look at the predictive accuracy of this multinomial logistic regression for the testing dataset.

```
set.seed(2)
inTraining <- createDataPartition(df1$AdoptionSpeed, p = .7, list = F)
training <- df1[inTraining, ]
testing  <- df1[-inTraining, ]

train_per_multinom <- multinom(AdoptionSpeed ~ ., data = training)
```

```
## # weights:  95 (72 variable)
## initial  value 13031.618777
## iter  10 value 11954.077344
## iter  20 value 11745.092759
## iter  30 value 11696.973699
## iter  40 value 11664.099690
## iter  50 value 11553.497682
## iter  60 value 11511.422233
## iter  70 value 11484.856314
## iter  80 value 11475.596771
## iter  80 value 11475.596729
## iter  80 value 11475.596729
## final  value 11475.596729
## converged
```

```
multinom_training <- training %>%
  mutate(fits = predict(train_per_multinom)) %>%
  mutate(multinom_accuracy = if_else(AdoptionSpeed == fits, 1, 0))
multinom_accuracy <- sum(multinom_training$multinom_accuracy==1)/nrow(multinom_training)
multinom_accuracy
```

```
## [1] 0.3450661
```

Test Accuracy:

```
multinom_test_pred <- predict(train_per_multinom, newdata = testing)
multinom_testing <- testing %>%
  mutate(AdoptionSpeed_pred = multinom_test_pred) %>%
  mutate(multinom_pred_accuracy = if_else(AdoptionSpeed == AdoptionSpeed_pred, 1, 0))

head(multinom_testing)
```

```
## # A tibble: 6 x 20
##    Type   Age Breed1 Breed2 Gender Color1 Color2 Color3 MaturitySize
##   <int> <int>  <int>  <int>  <int>  <int>  <int>  <int>        <int>
## 1     2     3    266      0      2      5      6      0            2
## 2     2    12    264    264      1      1      0      0            2
## 3     1     2    307      0      2      1      0      0            2
## 4     1    12    307      0      2      2      7      0            2
## 5     1    10    307    117      2      1      2      7            2
## 6     2    36    285    251      1      3      0      0            3
## # ... with 11 more variables: FurLength <int>, Vaccinated <int>,
## #   Dewormed <int>, Sterilized <int>, Health <int>, Fee <int>,
## #   VideoAmt <int>, PhotoAmt <dbl>, AdoptionSpeed <int>,
## #   AdoptionSpeed_pred <fct>, multinom_pred_accuracy <dbl>
```

```
multinom_accuracy_test <- sum(multinom_testing$multinom_pred_accuracy==1)/nrow(multinom_testing)
multinom_accuracy_test
```

```
## [1] 0.3587082
```

The testing accuracy generated by applying this multinomial logistic regression is only `0.3587`, which tells the overfitting happened with this logistic regression model.

LDA

```
set.seed(2)
inTraining <- createDataPartition(df1$AdoptionSpeed, p = .7, list = F)
training <- df1[inTraining, ]
testing  <- df1[-inTraining, ]

adop_lda <- lda(AdoptionSpeed ~ ., data = training)
fits <- predict(adop_lda)
confMat_lda <- table(fits$class, training$AdoptionSpeed)
confMat_lda
```

```
##
##        0    1    2    3    4
##   0    0    0    0    0    0
##   1   74  418  356  214  263
##   2   78  756  907  648  550
##   3   17  120  226  316  167
##   4   65  424  713  619 1166
```

Model Accuracy:

```
accuracy_lda <- sum(418, 907, 316, 1166)/sum(confMat_lda)
accuracy_lda
```

```
## [1] 0.3466716
```

Test Accuracy:

```
test_preds <- predict(adop_lda, newdata = testing)
confMat_lda_test <- table(test_preds$class, testing$AdoptionSpeed)
confMat_lda_test
```

```
##
```

```
##       0   1   2   3   4
##   0   0   0   0   0   0
##   1  30 178 168  79  91
##   2  32 328 377 252 221
##   3   8  51  93 158  69
##   4  27 164 323 266 553
```

```
accuracy_lda_test <- sum(178, 377, 158, 553)/sum(confMat_lda_test)
accuracy_lda_test
```

```
## [1] 0.3650519
```

Therefore, the test accuracy of this LDA model is `0.3651`, which is slightly overfitting.

QDA

```
adop_qda <- qda(AdoptionSpeed ~ ., data = training)
fits <- predict(adop_qda)
confMat_qda <- table(fits$class, training$AdoptionSpeed)
confMat_qda
```

```
##
##       0   1   2   3   4
##   0  14  21  31  17  20
##   1 109 831 706 486 519
##   2  61 514 840 583 479
##   3  14 128 225 326 168
##   4  36 224 400 385 960
```

Model Accuracy: this QDA model is better than LDA model.

```
accuracy_qda <- sum(14, 831, 840, 326, 960)/sum(confMat_qda)
accuracy_qda
```

```
## [1] 0.366926
```

Test Accuracy:

```
test_preds <- predict(adop_qda, newdata = testing)
confMat_qda_test <- table(test_preds$class, testing$AdoptionSpeed)
confMat_qda_test
```

```
##
##       0   1   2   3   4
##   0   2   5  17  11   6
##   1  41 342 320 173 222
##   2  21 235 345 275 223
##   3  11  53  94 136  64
##   4  22  86 185 160 419
```

```
accuracy_qda_test <- sum(2, 342, 345, 136, 419)/sum(confMat_qda_test)
accuracy_qda_test
```

```
## [1] 0.3587082
```

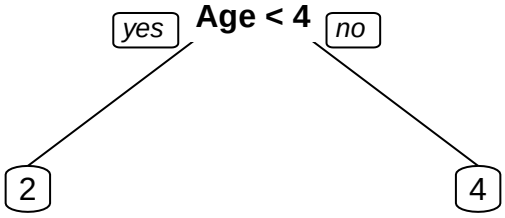
So, the test accuracy of this QDA model is `0.3587`, which is lower than the test accuracy of the LDA model but not overfitting.

Classification Tree

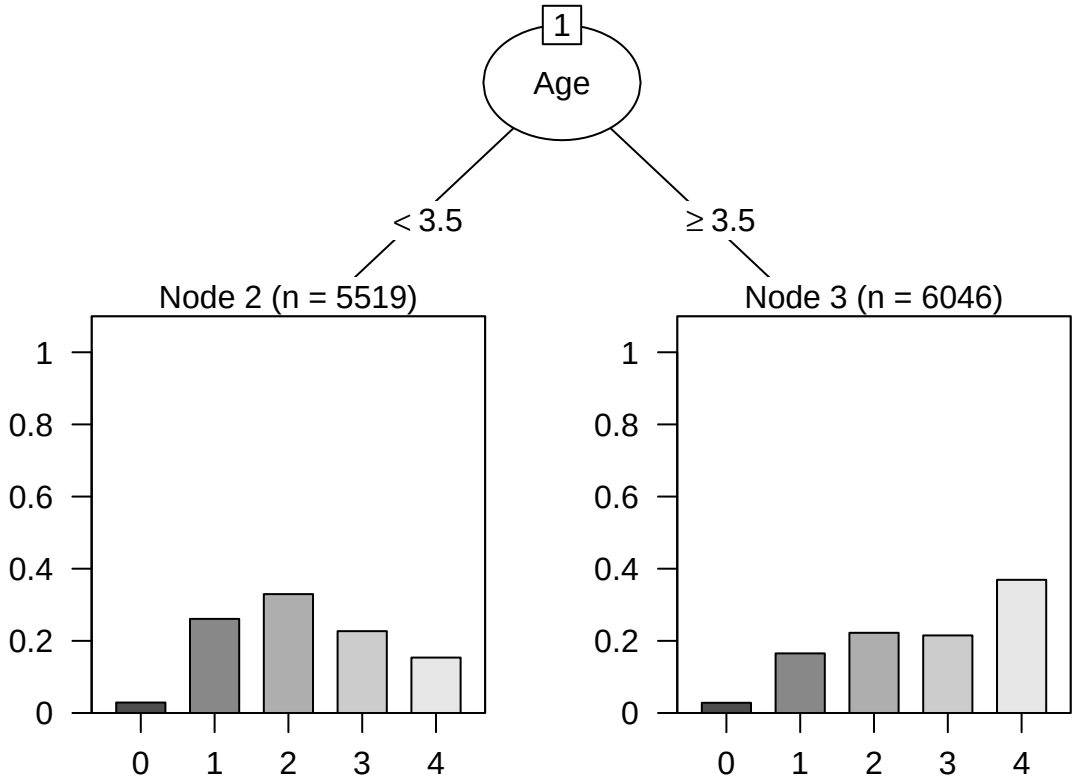
First, let run a basic decision tree

```
df2 <- df1 %>%
  mutate(AdoptionSpeed = as.factor(AdoptionSpeed))
```

```
adop_tree <- rpart(AdoptionSpeed ~ . , df2)
prp(adop_tree)
```




```
plot(as.party(adop_tree))
```



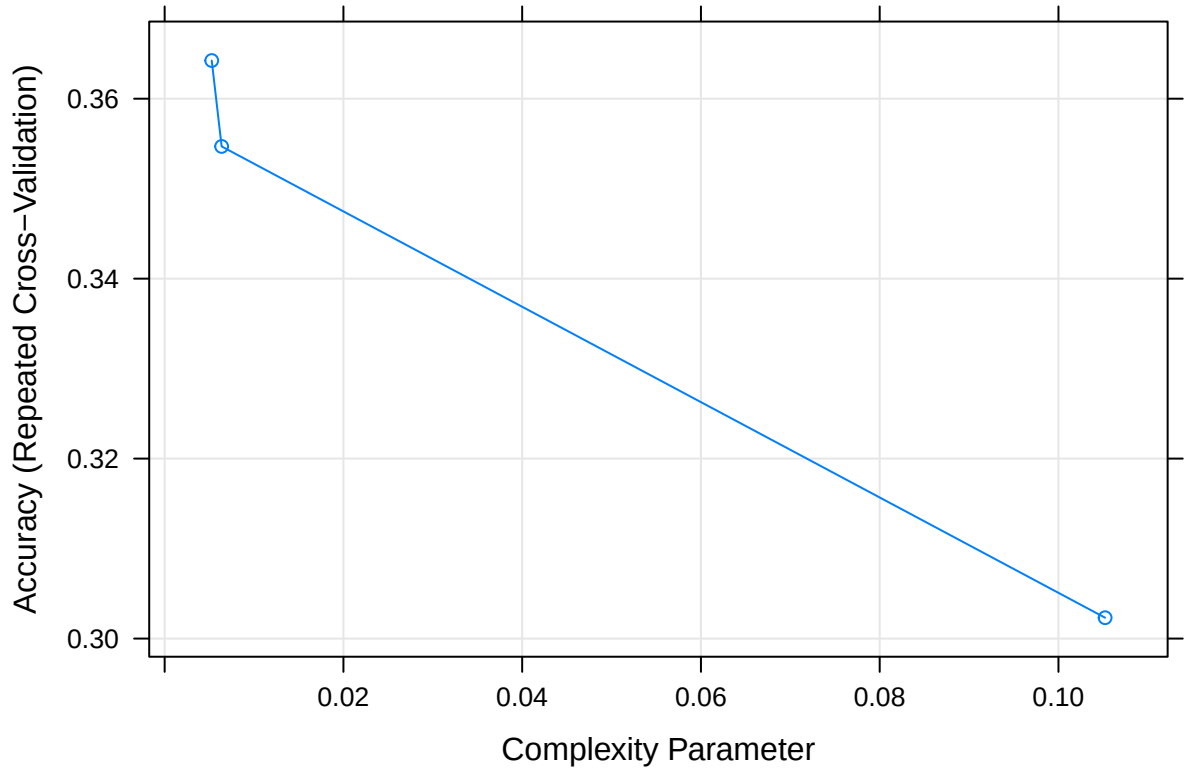

```
adop_tree
```

```
## n= 11565
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 11565 8402 2 (0.029 0.21 0.27 0.22 0.27)
##   2) Age< 3.5 5519 3700 2 (0.029 0.26 0.33 0.23 0.15) *
##   3) Age>=3.5 6046 3814 4 (0.028 0.17 0.22 0.22 0.37) *
```

```
printcp(adop_tree)
```

```
##
## Classification tree:
## rpart(formula = AdoptionSpeed ~ ., data = df2)
##
## Variables actually used in tree construction:
## [1] Age
##
## Root node error: 8402/11565 = 0.7265
##
## n= 11565
##
##         CP nsplit rel error  xerror      xstd
## 1 0.10569      0   1.00000 1.00000 0.0057054
## 2 0.01000      1   0.89431 0.89431 0.0061061
```

```r
set.seed(2)
inTraining <- createDataPartition(df2$AdoptionSpeed, p = .7, list = F)
training <- df2[inTraining, ]
testing  <- df2[-inTraining, ]
fit_control <- trainControl(method = "repeatedcv",
                            number = 10,
                            repeats = 10)
cv_adop_tree <- train(AdoptionSpeed ~ .,
                      data = training,
                      method = "rpart",
                      trControl = fit_control)
plot(cv_adop_tree)
```

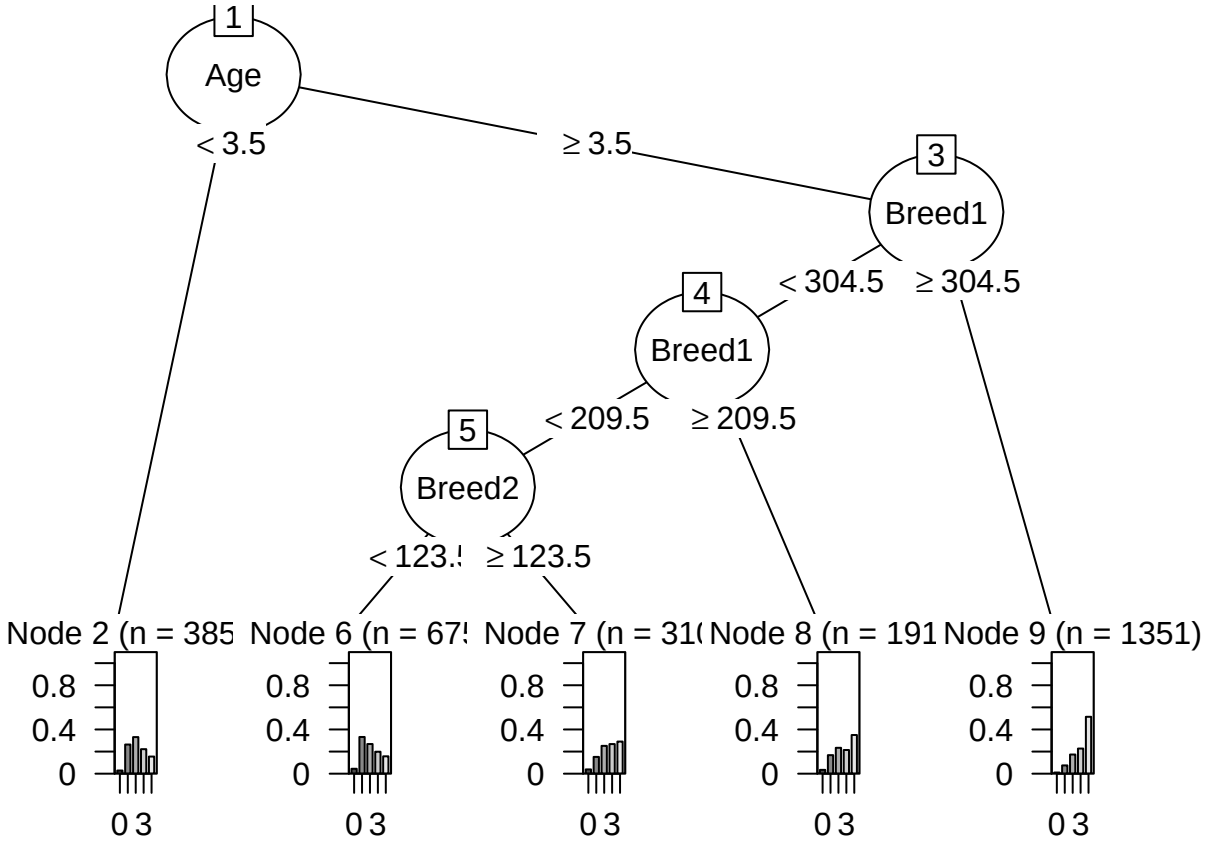



```r
cv_adop_tree #cp: method for choosing final nodes
```

```
## CART
##
## 8098 samples
##   17 predictor
##    5 classes: '0', '1', '2', '3', '4'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 7287, 7288, 7288, 7288, 7290, 7289, ...
## Resampling results across tuning parameters:
##
```

```
##   cp           Accuracy   Kappa
##   0.005269420  0.3642391  0.13550474
##   0.006374299  0.3546948  0.11949706
##   0.105218426  0.3023170  0.04170737
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.00526942.
```

```
plot(as.party(cv_adop_tree$finalModel))
```




```
tree_test_pred <- predict(cv_adop_tree, newdata = testing)
confMat_tree <- table(tree_test_pred, testing$AdoptionSpeed)
confMat_tree
```

```
##
## tree_test_pred   0   1   2   3   4
##              0   0   0   0   0   0
##              1  17  89  89  66  35
##              2  50 424 544 398 251
##              3   0   0   0   0   0
##              4  32 218 315 301 638
```

```
accuracy_tree_test <- sum(89, 544, 638)/sum(confMat_tree)
accuracy_tree_test
```

```
## [1] 0.3665994
```

The predictive accuracy on testing dataset of this decision tree is `0.3666`, which is higher than the logistic

regression model but still too low.

Now, let's try bagging and random forest for finding the best decision tree ####Bagging

```
set.seed(2)
inTraining <- createDataPartition(df2$AdoptionSpeed, p = .7, list = F)
training <- df2[inTraining, ]
testing  <- df2[-inTraining, ]
```

```
set.seed(10982)
adop_bag <- randomForest(AdoptionSpeed ~ ., data = training, mtry = 17) # mtry:  the number of predicto
adop_bag         # default setting of the number of tree is 500, which is pretty enough.
```

```
##
## Call:
##  randomForest(formula = AdoptionSpeed ~ ., data = training, mtry = 17)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
##         OOB estimate of  error rate: 61.2%
## Confusion matrix:
##    0   1   2   3    4 class.error
## 0 21  71  45  41   54   0.9094828
## 1 15 610 532 247  304   0.6428571
## 2  5 503 782 435  490   0.6469526
## 3  4 297 513 499  474   0.7207611
## 4 12 235 404 275 1230   0.4294991
```

```
accuracy_bag <- 1 - .612
accuracy_bag
```

```
## [1] 0.388
```

Test Accuracy:

```
test_preds <- predict(adop_bag, newdata = testing)
test_df_bag <- testing %>%
  mutate(y_hat_bag = test_preds,
         accuracy = if_else(y_hat_bag==AdoptionSpeed,1,0))
accuracy_bag_test <- sum(test_df_bag$accuracy==1)/nrow(test_df_bag)
accuracy_bag_test
```

```
## [1] 0.3827517
```

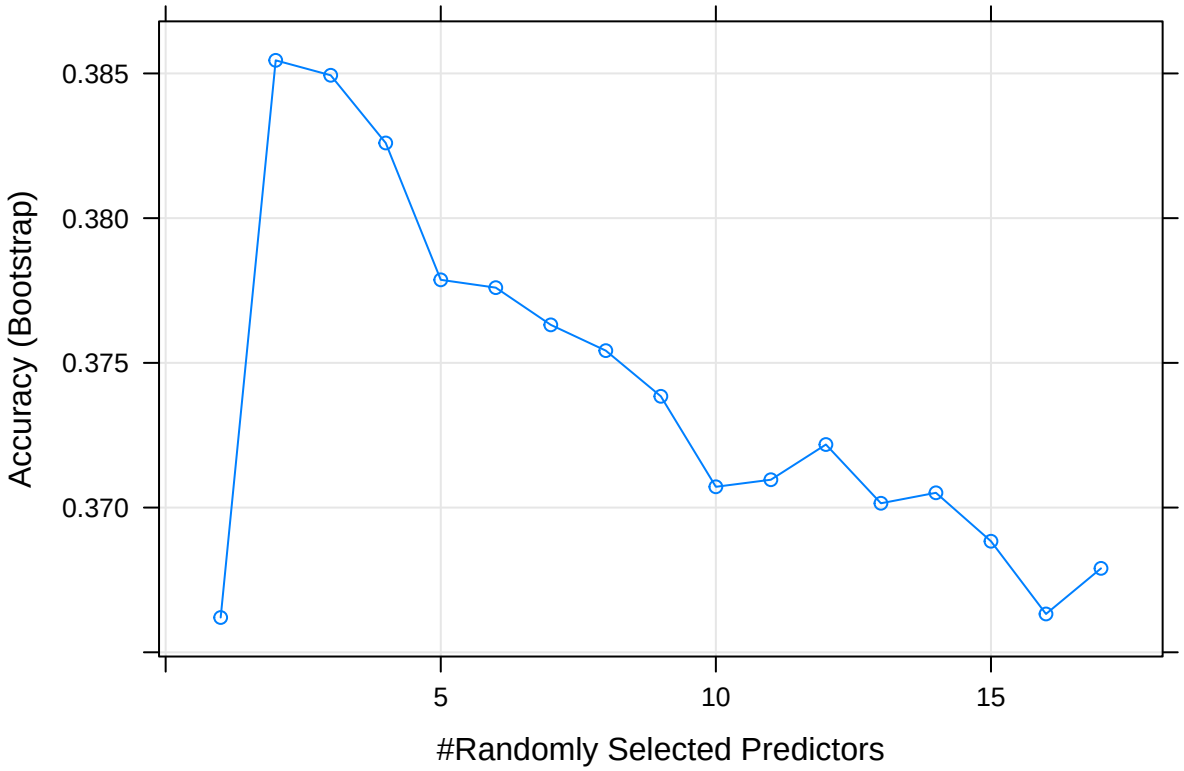
The test accuracy of this bagging model is 0.3828.

Random Forest

```
set.seed(1982)

rf_adop_cv <- train(AdoptionSpeed ~ .,
                    data = training,
                    method = "rf",
                    ntree = 100,
                    importance = T, # output the 'importance'
                    tuneGrid = data.frame(mtry = 1:17)) # important piece for tree function training
rf_adop_cv
```

```
## Random Forest
##
## 8098 samples
##   17 predictor
##    5 classes: '0', '1', '2', '3', '4'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8098, 8098, 8098, 8098, 8098, 8098, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.3662017  0.1355399
##    2    0.3854519  0.1755899
##    3    0.3849365  0.1801150
##    4    0.3825965  0.1788176
##    5    0.3778688  0.1732434
##    6    0.3775986  0.1733418
##    7    0.3763124  0.1718491
##    8    0.3754225  0.1709925
##    9    0.3738438  0.1690504
##   10    0.3707178  0.1649291
##   11    0.3709622  0.1654931
##   12    0.3721782  0.1673342
##   13    0.3701488  0.1647211
##   14    0.3705111  0.1652672
##   15    0.3688365  0.1631108
##   16    0.3663246  0.1599850
##   17    0.3678998  0.1621060
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(rf_adop_cv)
```



Therefore, for highest accuracy of prediction, the tree with `mtry = 2` is chosen as the best tree model.

```
set.seed(1982)
rf_adop_2 <- randomForest(AdoptionSpeed ~.,
                          data = training,
                          mtry = 2)
rf_adop_2
```

```
##
## Call:
##  randomForest(formula = AdoptionSpeed ~ ., data = training, mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 60.56%
## Confusion matrix:
##    0   1    2   3    4 class.error
## 0 11  76   67  15   63   0.9525862
## 1  2 446  830  78  352   0.7388759
## 2  0 415 1060 166  574   0.5214447
## 3  0 216  699 265  607   0.8517068
## 4  1 173  477  93 1412   0.3450835
```

```
accuracy_2 <- 1- .6056
accuracy_2
```

```
## [1] 0.3944
```

Test Accuracy:

```
test_preds <- predict(rf_adop_2, newdata = testing)
test_df_rf <- testing %>%
  mutate(y_hat_rf = test_preds,
         accuracy = if_else(y_hat_rf==AdoptionSpeed,1,0))
accuracy_rf_2 <- sum(test_df_rf$accuracy==1)/nrow(test_df_rf)
accuracy_rf_2
```

```
## [1] 0.4012114
```

After comparing the test accuracy of bagging and random forest with `mtry=2`, the random forest model get better test accuracy but is slightly overfitting which could be accepted.