

Titanic - Machine Learning from Disaster

Sangkon Han(sangkon@pusan.ac.kr)

2023-04-05

Contents

준비작업	2
multiplot() function generation	2
데이터 불러오기	3
데이터 확인 및 변환	3
변수 의미 설명	3
변수 속성 변환	4
탐색적 데이터 분석	4
수치값을 활용한 data 확인	4
str()	4
summary()	5
결측치 처리	6
VIM 패키지를 사용	6
tidyverse 패키지를 활용	7
Age	10
Pclass	11
Fare	12
Sex	14
데이터 전처리 및 특징 확인	16
Age -> Age.Group	16
SibSp & Parch -> FamilySized	16
2.4 Name & Sex -> title	17
Ticket -> ticket.size	18
Embarked	20
Fare	20
Survived와 관련된 특징 선택	20
데이터셋 분리	20
Pclass	20
Sex	21
Embarked	22
FamilySized	23
Age.Group	24
title	25
ticket.size	26
특징 추출	27
머신러닝 모델	28
랜덤포레스트	28

Feature importance check	28
예측 및 submit 파일 작성	29

Kaggle의 대표적인 Competition 중 하나인 Titanic 생존자 예측에 관한 내용을 다루고 있습니다.
처음 Kaggle에 도전하시는 분들이 참고하실만한 자료가 되었으면 합니다.

준비작업

pkg <- c("readr", "descr", "VIM", "RColorBrewer", "scales", "tidyverse", "randomForest", "caret", "ROCR", "scater")와 같이 설치할 패키지를 선택하고, install.packages(pkg)를 사용해서 패키지를 설치하세요.

```
library(readr)
library(descr)
library(VIM)
library(ggplot2)
library(RColorBrewer)
library(scales)
library(dplyr)
library(purrr)
library(tidyr)
library(randomForest)
library(caret)
library(ROCR)
```

multiplot() function generation

```
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols: Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
#
multiplot <- function(..., plotlist = NULL, file, cols = 1, layout = NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                    ncol = cols, nrow = ceiling(numPlots/cols))
  }
}
```

```

if (numPlots==1) {
  print(plots[[1]])

} else {
  # Set up the page
  grid.newpage()
  pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout)),

# Make each plot, in the correct location
for (i in 1:numPlots) {
  # Get the i,j matrix positions of the regions that contain this subplot
  matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

  print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                    layout.pos.col = matchidx$col))
}
}
}

```

데이터 불러오기

titanic competition에서는 Model을 생성하는데 사용하는 train data와 실제 예측(추정)에 사용하는 test data가 분리되어 있습니다. 여기서는 저 2개 data들을 불러와서 하나로 묶을 것 입니다. 따로 분리되어 있는 데이터들을 하나로 묶는 이유는 모델링에 사용되는 입력변수들을 Feature engineering, Pre-processing 할 때 동일하게 작업하기 위해서입니다.

```

train <- readr::read_csv('./data/titanic_train.csv')
test <- readr::read_csv('./data/titanic_test.csv')
full <- dplyr::bind_rows(train, test)

```

- `read.csv()` 대신 `readr` 패키지의 `read_csv()`를 사용한 이유는 `read_csv()`가 조금 더 빠름, 하지만 문자열(Character)과 요인 변수(Factor)를 구별하지 못하고 모두 Chr속성으로 처리하기 때문에 주의를 요함
- 추가로 두 데이터를 `full`로 합칠때도 `rbind()`를 사용하지 않은 이유는 `test`에는 Titanic competition 의 종속변수(타겟변수, Y)인 `Survived`가 존재하지 않기 때문에 데이터의 차원(dimension)이 맞지않아 `rbind()`로는 병합되지 않기 때문에 `dplyr::bind_rows()`를 사용하면 `test`의 `Survived`를 NA로 처리하면서 하나로 병합

데이터 확인 및 변환

변수 의미 설명

변수명	해석(의미)	Type
PassengerID	승객을 구별하는 고유 ID number	Int
Survived	승객의 생존 여부를 나타내며 생존은 1, 사망은 0 입니다.	Factor
Pclass	선실의 등급으로서 1등급(1)부터 3등급(3)까지 3개 범주입니다.	Ord.Factor
Name	승객의 이름	Factor
Sex	승객의 성별	Factor
Age	승객의 나이	Numeric
SibSp	각 승객과 동반하는 형제 또는 배우자의 수를 설명하는 변수이며 0부터 8까지 존재합니다.	Integer

변수명	해석(의미)	Type
Parch	각 승객과 동반하는 부모님 또는 자녀의 수를 설명하는 변수이며 0부터 9까지 존재합니다.	Integer
Ticket	승객이 탑승한 티켓에 대한 문자열 변수	Factor
Fare	승객이 지금까지 여행하면서 지불한 금액에 대한 변수	Numeric
Cabin	각 승객의 선실을 구분하는 변수이며 범주와 결측치가 너무 많습니다.	Factor
Embarked	승선항, 출항지를 나타내며 C, Q, S 3개 범주이다.	Factor

변수 속성 변환

앞서 설명한 문제로 인해서 몇가지 변수 속성을 처리하도록 하겠습니다.

```
full <- full %>%
  dplyr::mutate(Survived = factor(Survived),
    Pclass = factor(Pclass, ordered = T),
    Name = factor(Name),
    Sex = factor(Sex),
    Ticket = factor(Ticket),
    Cabin = factor(Cabin),
    Embarked = factor(Embarked))
```

탐색적 데이터 분석

Data가 어떻게 구성이 되어 있고 그 안에 결측치(Missing value) 혹은 이상치(Outlier)가 존재하는지 등등 원시 데이터(Raw data)에 대해 탐색하고 이해하는 과정입니다. 여기서는 다양한 function들과 시각화(Visualization)을 사용할 것입니다.

수치값을 활용한 data 확인

가장 먼저 head(), summary() 등 다양한 함수들의 결과값(Output)들을 통해 data를 확인하도록 하겠습니다.

```
head(full, 10)
```

```
## # A tibble: 10 x 12
##   PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##       <dbl>   <fct>   <ord>  <fct> <fct> <dbl> <dbl> <dbl> <fct> <dbl> <fct>
## 1         1      0      3  Braun~ male    22     1     0 A/5 2~  7.25 <NA>
## 2         2      1      1  Cumin~ fema~   38     1     0 PC 17~  71.3  C85
## 3         3      1      3  Heikk~ fema~   26     0     0 STON/~  7.92 <NA>
## 4         4      1      1  Futre~ fema~   35     1     0 113803 53.1  C123
## 5         5      0      3  Allen~ male    35     0     0 373450  8.05 <NA>
## 6         6      0      3  Moran~ male    NA     0     0 330877  8.46 <NA>
## 7         7      0      1  McCar~ male    54     0     0 17463   51.9  E46
## 8         8      0      3  Palss~ male    2       3     1 349909 21.1  <NA>
## 9         9      1      3  Johns~ fema~   27     0     2 347742 11.1  <NA>
## 10        10     1      2  Nasse~ fema~  14     1     0 237736 30.1  <NA>
## # ... with 1 more variable: Embarked <fct>
```

약간의 결측치가 있음을 확인할 수 있다.

```
str()
```

```
str(full)
```

```

## # tibble [1,309 x 12] (S3: tbl_df/tbl/data.frame)
## $ PassengerId: num [1:1309] 1 2 3 4 5 6 7 8 9 10 ...
## $ Survived : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
## $ Pclass   : Ord.factor w/ 3 levels "1"<"2"<"3": 3 1 3 1 3 3 1 3 3 2 ...
## $ Name     : Factor w/ 1307 levels "Abbing, Mr. Anthony",...: 156 287 531 430 23 826 775 922 613 8...
## $ Sex      : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 1 1 ...
## $ Age      : num [1:1309] 22 38 26 35 35 NA 54 2 27 14 ...
## $ SibSp    : num [1:1309] 1 1 0 1 0 0 0 3 0 1 ...
## $ Parch    : num [1:1309] 0 0 0 0 0 0 0 1 2 0 ...
## $ Ticket   : Factor w/ 929 levels "110152","110413",...: 721 817 915 66 650 374 110 542 478 175 ...
## $ Fare     : num [1:1309] 7.25 71.28 7.92 53.1 8.05 ...
## $ Cabin    : Factor w/ 186 levels "A10","A11","A14",...: NA 107 NA 71 NA NA 164 NA NA NA ...
## $ Embarked : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...

```

Train, test data를 합쳐서 전체 관측치(레코드, 행, Row)는 총 1309개(train : 891, test : 418)이고 변수(열, Feature, variable, column)는 12개 임을 알 수 있습니다. 그 외에도 각 변수들의 속성이 무엇인지와 factor 속성인 변수들에 범주가 몇 개인지까지 알 수 있습니다. 또한 head()에서는 Age에만 존재하는 줄 알았던 결측치(NA)가 Cabin을 비롯한 다른 변수들에도 있음을 알 수 있습니다.

`summary()`

```
summary(full)
```

	PassengerId	Survived	Pclass	Name	
## Min.	: 1	0 :549	1:323	Connolly, Miss. Kate : 2	
## 1st Qu.:	328	1 :342	2:277	Kelly, Mr. James : 2	
## Median :	655	NA's:418	3:709	Abbing, Mr. Anthony : 1	
## Mean :	655			Abbott, Master. Eugene Joseph : 1	
## 3rd Qu.:	982			Abbott, Mr. Rossmore Edward : 1	
## Max. :	1309			Abbott, Mrs. Stanton (Rosa Hunt): 1	
##			(Other)	:1301	
	Sex	Age	SibSp	Parch	Ticket
## female:	466	Min. : 0.17	Min. :0.0000	Min. :0.000	CA. 2343: 11
## male :	843	1st Qu.:21.00	1st Qu.:0.0000	1st Qu.:0.000	1601 : 8
##		Median :28.00	Median :0.0000	Median :0.000	CA 2144 : 8
##		Mean :29.88	Mean :0.4989	Mean :0.385	3101295 : 7
##		3rd Qu.:39.00	3rd Qu.:1.0000	3rd Qu.:0.000	347077 : 7
##		Max. :80.00	Max. :8.0000	Max. :9.000	347082 : 7
##		NA's :263		(Other) :1261	
	Fare	Cabin	Embarked		
## Min. :	0.000	C23 C25 C27 : 6	C :270		
## 1st Qu.:	7.896	B57 B59 B63 B66: 5	Q :123		
## Median :	14.454	G6 : 5	S :914		
## Mean :	33.295	B96 B98 : 4	NA's: 2		
## 3rd Qu.:	31.275	C22 C26 : 4			
## Max. :	512.329	(Other) : 271			
## NA's :	1	NA's :1014			

`summary()`는 data에 대한 많은 정보를 제공해줍니다. 수량형(Integer, Numeric) 변수들의 대푯값들, 범주형(Factor) 변수들의 범주가 몇 개인지 그리고 각 범주에 속하는 관측치의 갯수들까지 모두 수치값들로 보여줍니다.

여기서 확인하고 넘어갈 것들은 다음과 같습니다.

1. `Survived` : 이번 competition의 타겟 변수이며 418개의 결측치는 Test data 때문입니다.
2. `Pclass` : 1등급, 2등급, 3등급으로 범주가 세 개인데 3등급 승객이 가장 많습니다.

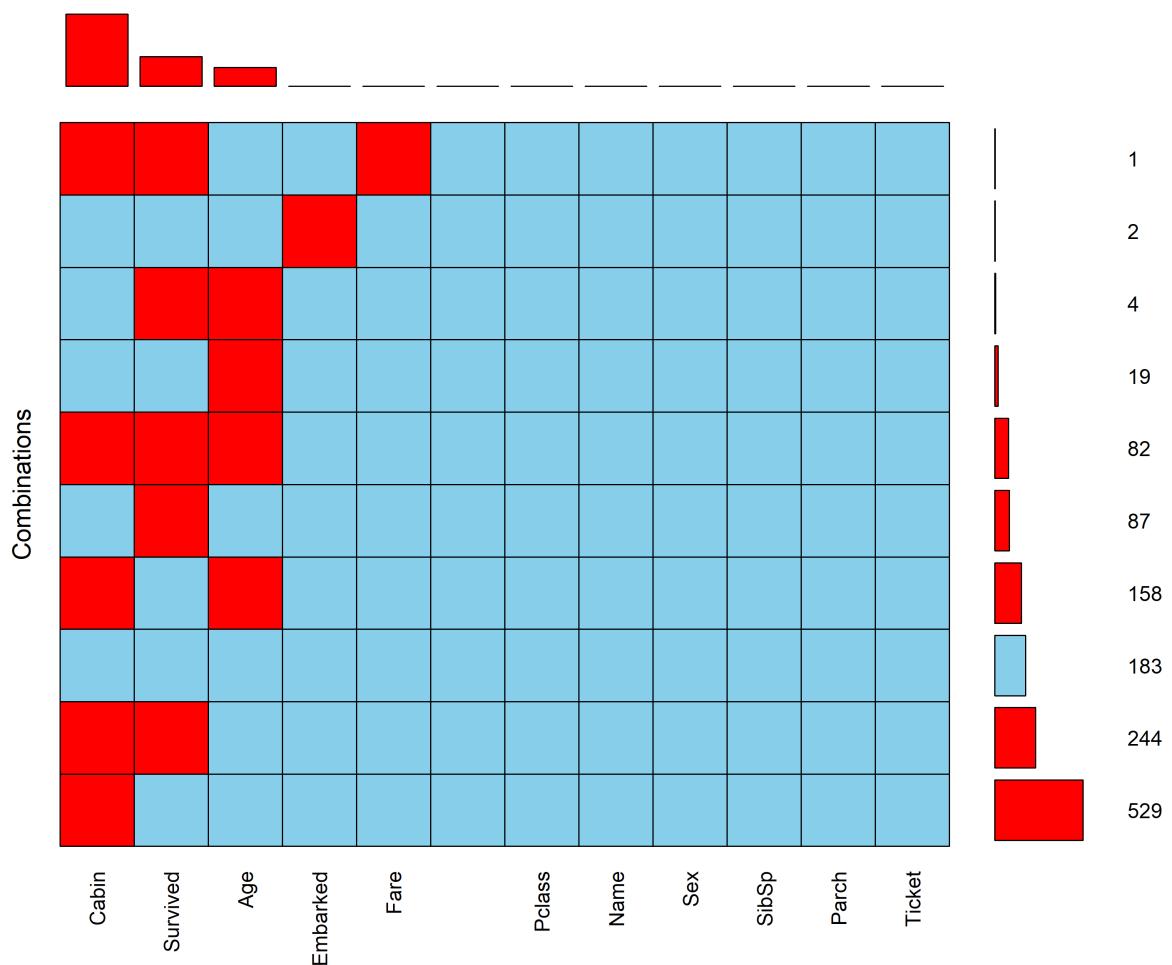
3. **Name** : 이름이 비슷한 사람들이 있습니다. 따라서 혼자 탄 승객도 있지만 가족들과 같이 탑승한 승객도 있음을 알 수 있습니다.
4. **Sex** : 남성이 여성보다 2배 가까이 더 많습니다.
5. **Age** : 0.17부터 80세까지 있는데 17을 잘못 기입한 이상치 인건지 확인이 필요해보이며 263개의 결측치가 존재합니다.
6. **SibSp** : 0부터 8까지 있는데 3분위수가 1이므로 부부 혹은 형제와 함께 Titanic호에 탑승했음을 알 수 있습니다.
7. **Parch** : 0부터 9까지 있지만 3분위수가 0인 것을 통해서 부모, 자녀들과 함께 탄 승객들이 거의 없음을 알 수 있습니다. **SibSp**와 **Parch**는 모두 가족관계를 나타내는 변수들입니다. 이를 통해서 가족 중 누가 탔는지는 모르지만 동승한 인원이 총 몇 명인지 구하고, 그것을 바탕으로 가족의 규모를 나타내는 **FamilySized**라는 범주형 파생변수를 만들 것입니다.
8. **Ticket** : 3.1.2 `str()`의 결과와 같이보면 완전히 동일한 티켓을 가진 승객들도 있고, 티켓이 일정 부분만 겹치는 승객들도 있고 아예 다른 티켓을 가진 승객이 있음을 알 수 있습니다. 이것을 이용해서 `ticket.size`라는 파생변수를 만들 계획입니다.
9. **Fare** : 0부터 512까지 있고 1개의 결측치가 있습니다. 3분위수가 31.275이며 최댓값이 512인게 신경쓰입니다.
10. **Cabin** : 총 12개 Feature들 중에서 가장 많은(1014개) 결측치를 갖고 있습니다. 배의 구역을 나타내는 변수인데 활용할 방법이 없다면 버려야 될 것 같습니다.
11. **Embarked** : 총 3개 범주로 구성이 되어있고 S가 가장 많으며 2개의 결측치가 있습니다.

결측치 처리

위에서 언급했던 결측치가 존재하는 변수는 무엇이고 또 얼마나 존재하는지 확인해보는 과정입니다. `dplyr`, `ggplot2`, `VIM` 패키지들을 이용해서 수치적으로 확인하면서 동시에 시각적으로도 확인해보려고 합니다.

VIM 패키지를 사용

```
VIM:::aggr(full, prop = FALSE, combined = TRUE, numbers = TRUE,
           sortVars = TRUE, sortCombs = TRUE)
```



```
##
## Variables sorted by number of missings:
##      Variable Count
##      Cabin    1014
##      Survived   418
##      Age     263
##      Embarked    2
##      Fare      1
##      PassengerId  0
##      Pclass     0
##      Name      0
##      Sex       0
##      SibSp     0
##      Parch     0
##      Ticket     0
```

tidyverse 패키지를 활용

VIM 패키지를 이용해서 한 번에 결측치를 확인하는 것 외에 tidyverse에 존재하는 다양한 패키지들을 이용하여 결측치를 확인하는 방법들입니다. 먼저 dplyr로 각 변수별 결측치 비율을 구하는 것입니다.

```
full %>%
  dplyr::summarize_all(funs(sum(is.na(.))/n()))
```

```

## # A tibble: 1 x 12
##   PassengerId Survived Pclass  Name   Sex   Age SibSp Parch Ticket     Fare Cabin
##       <dbl>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1          0     0.319     0     0    0.201     0     0     0 7.64e-4 0.775
## # ... with 1 more variable: Embarked <dbl>

```

이렇게 변수들에 존재하는 결측치의 비율을 확인하는 방법도 있지만, 시각자료들을 이용해서도 확인할 수 있습니다. 아래의 2개 Bar plot들을 봄주시기 바랍니다.

```

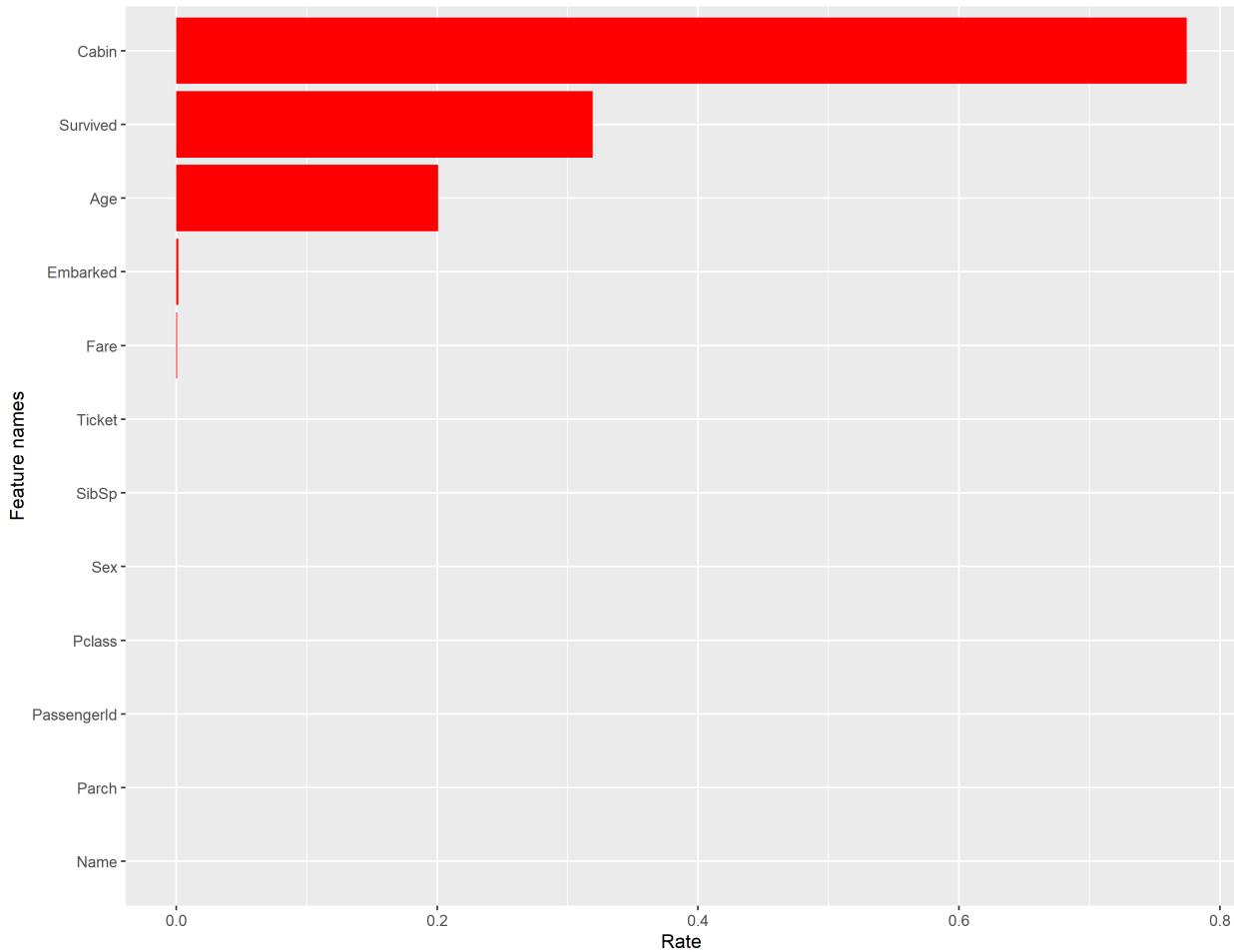
missing_values <- full %>%
  dplyr::summarize_all(funs(sum(is.na(.))/n()))

missing_values <- tidyverse::gather(missing_values,
                                     key = "feature", value = "missing_pct")

missing_values %>%
  ggplot(aes(x = reorder(feature, missing_pct), y = missing_pct)) +
  geom_bar(stat = "identity", fill = "red") +
  ggtitle("Rate of missing values in each features") +
  theme(plot.title = element_text(face = "bold",      # 글씨체
                                  hjust = 0.5,      # Horizon(가로비율) = 0.5
                                  size = 15, color = "darkblue")) +
  labs(x = "Feature names", y = "Rate") +
  coord_flip()

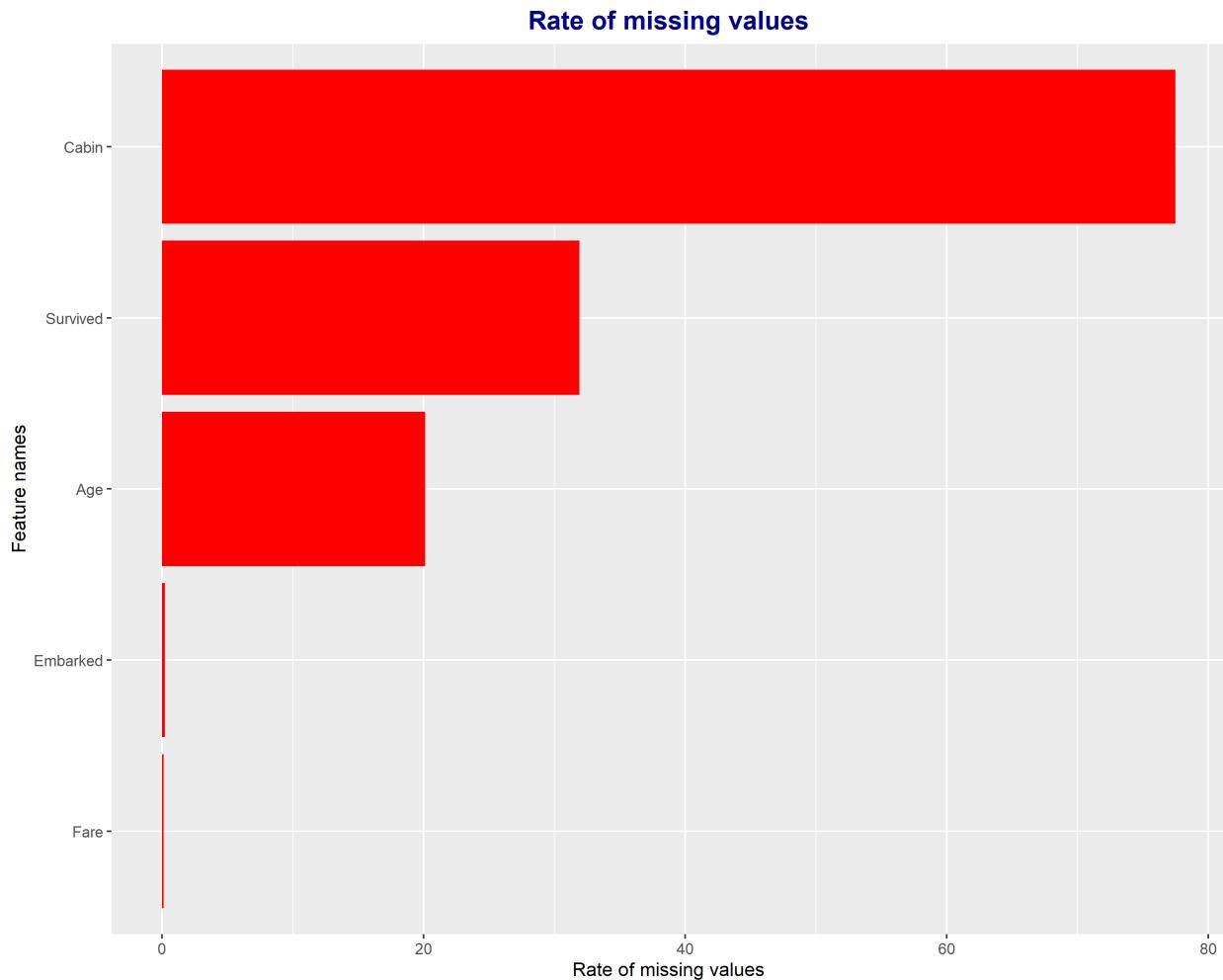
```

Rate of missing values in each features



위 막대 그래프를 보시면 모든 feature들에 대해 결측치 비율을 확인 할 수 있습니다. 하지만 실질적으로 우리가 궁금한 것은 결측치가 존재하는 변수는 무엇인지와 그 안에 얼마나 되는 결측치가 존재하느냐입니다. 때문에 purrr 패키지를 이용해서 결측치 비율을 계산한 후, 하나라도 존재하는 변수만 추출한 뒤에 시각화 해보았습니다.

```
miss_pct <- purrr::map_dbl(full, function(x){round((sum(is.na(x))/length(x)) * 100, 1) })
miss_pct <- miss_pct[miss_pct > 0]
data.frame(miss = miss_pct, var = names(miss_pct), row.names = NULL) %>%
  ggplot(aes(x = reorder(var, miss), y = miss)) +
  geom_bar(stat = 'identity', fill = 'red') +
  ggttitle("Rate of missing values") +
  theme(plot.title = element_text(face = "bold",      # 글씨체
                                  hjust = 0.5,      # Horizon(가로비율) = 0.5
                                  size = 15, color = "darkblue")) +
  labs(x = 'Feature names', y = 'Rate of missing values') +
  coord_flip()
```



이를 통해 총 12개 변수들 중에서 4개 변수에만 결측치가 존재하며(Survived는 test data 때문이므로 제외) Cabin, Age, Embarked, Fare 순으로 결측치가 많음을 알 수 있습니다.

Age

```
age.p1 <- full %>%
  ggplot(aes(Age)) +
  # 히스토그램 그리기, 설정
  geom_histogram(breaks = seq(0, 80, by = 1), # 간격 설정
                 col    = "red",                # 막대 경계선 색깔
                 fill   = "green",               # 막대 내부 색깔
                 alpha  = .5) +                # 막대 투명도 = 50%
  # Plot title
  ggtitle("All Titanic passengers age histogram") +
  theme(plot.title = element_text(face = "bold",      # 글씨체
                                   hjust = 0.5,       # Horizon(가로비율) = 0.5
                                   size = 15, color = "darkblue"))

age.p2 <- full %>%
  # test data set의 Survived == NA 인 값을 제외
  filter(!is.na(Survived)) %>%
  ggplot(aes(Age, fill = Survived)) +
```

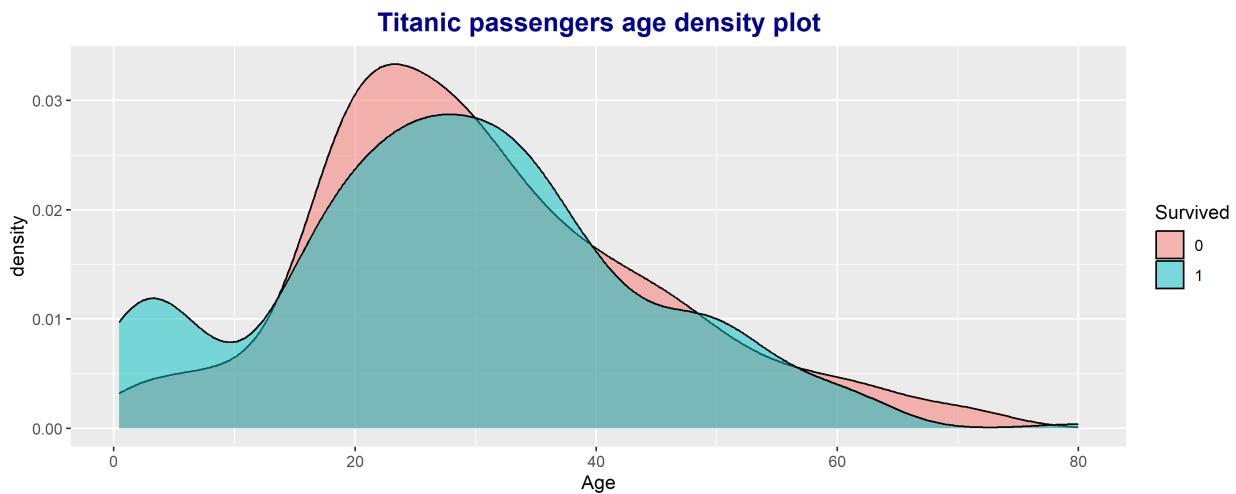
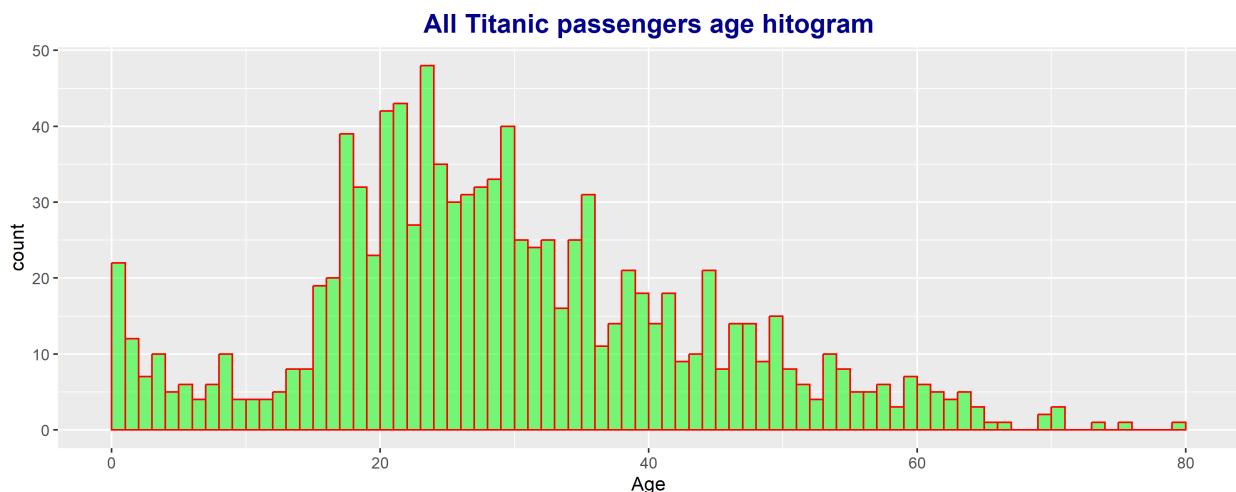
```

geom_density(alpha = .5) +
ggtitle("Titanic passengers age density plot") +
theme(plot.title = element_text(face = "bold", hjust = 0.5,
                                size = 15, color = "darkblue"))

# multiplot layout 형식 지정
multi.layout = matrix(c(1, 1, 2, 2), 2, 2, byrow = T)

# 위에서 생성한 2개의 그래프 한 화면에 출력
multiplot(age.p1, age.p2, layout = multi.layout)

```



Pclass

각 Pclass에 해당하는 탑승객의 빈도수를 시각화 해보겠습니다. dplyr 패키지를 활용해서 Pclass별로 집단화(그룹핑) 시킨 후 범주별 빈도수를 나타내는 Data Frame을 생성한 뒤에 ggplot으로 시각화 했습니다.

```

full %>%
  # dplyr::group_by(), summarize() 를 이용해서 Pclass 빈도수 구하기
  group_by(Pclass) %>%
  summarize(N = n()) %>%
  # Aesthetic setting
  ggplot(aes(Pclass, N)) +

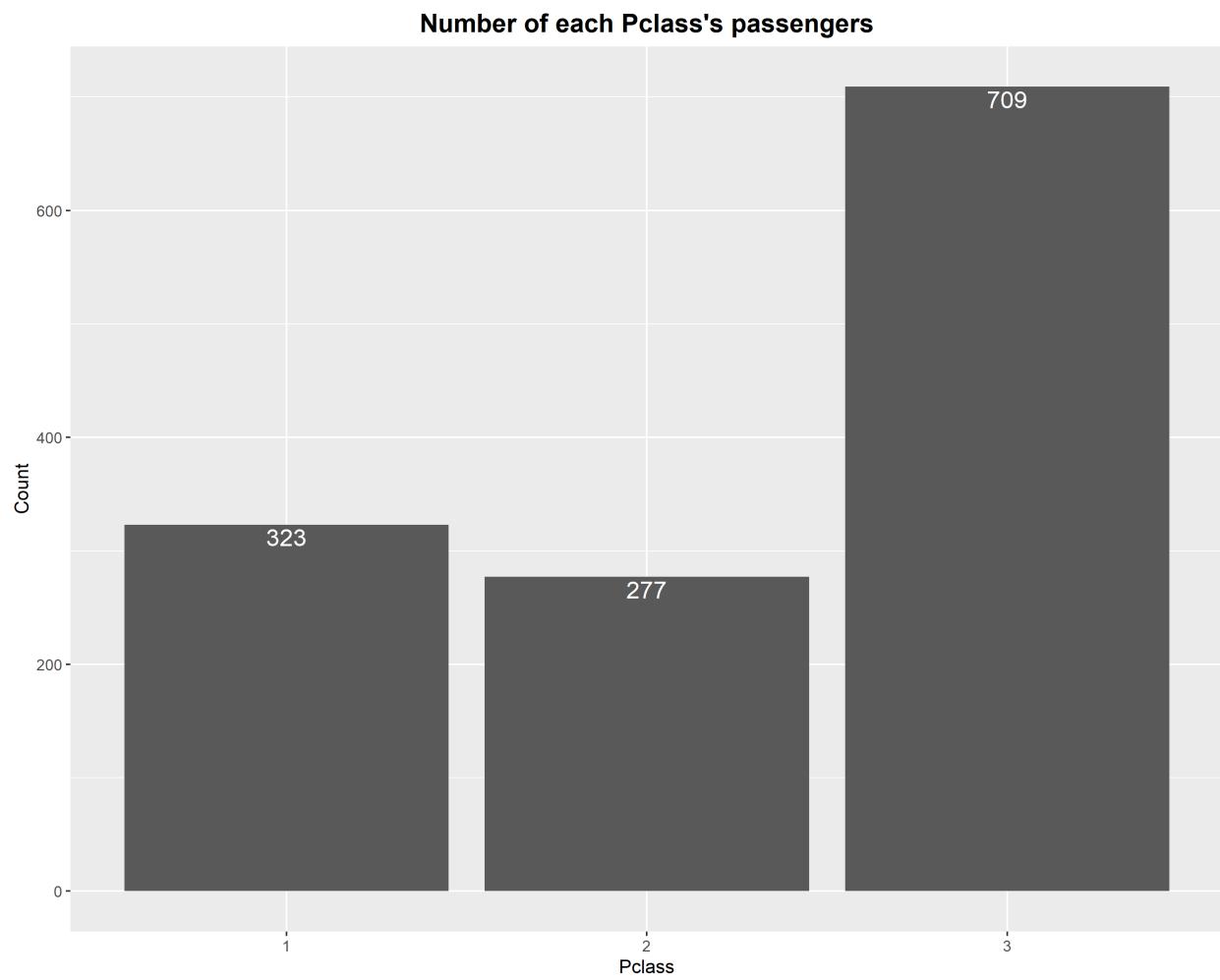
```

```

geom_col() +
# Pclass 빈도수 plot에 출력
geom_text(aes(label = N),           # Plot의 y에 해당하는 N(빈도수)를 맵핑
          size = 5,                # 글씨 크기
          vjust = 1.2,              # vertical(가로) 위치 설정
          color = "#FFFFFF") +    # 글씨 색깔 : 흰색

# Plot title
ggtitle("Number of each Pclass's passengers") +
# Title setting
theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15)) +
# x, y axis name change
labs(x = "Pclass", y = "Count")

```



3등급 객실에 탑승한 승객이 가장 많음을 알 수 있습니다. 객실의 등급과 생존율이 연관있는지는 Chapter5에서 다루도록 하겠습니다.

Fare

탑승객이 지불한 금액을 나타내는 Fare 변수에 대한 시각화입니다.

히스토그램과 상자그림 두 개를 이용했습니다.

```

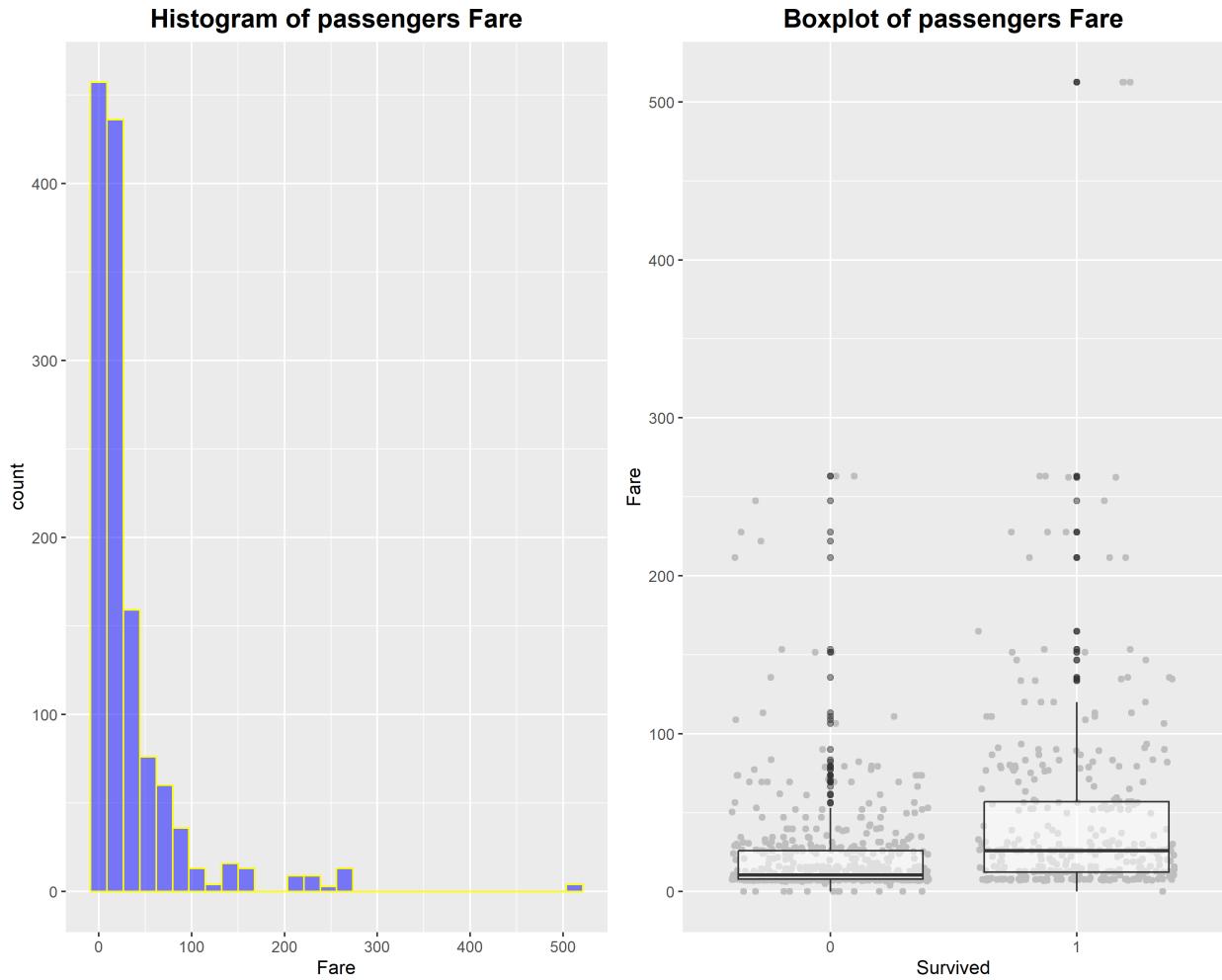
# Histogram
Fare.p1 <- full %>%
  ggplot(aes(Fare)) +
  geom_histogram(col     = "yellow",
                 fill    = "blue",
                 alpha   = .5) +
  ggtitle("Histogram of passengers Fare") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15))

# Boxplot
Fare.p2 <- full %>%
  filter(!is.na(Survived)) %>%
  ggplot(aes(Survived, Fare)) +
  # 관측치를 회색점으로 찍되, 중복되는 부분은 퍼지게 그려줍니다.
  geom_jitter(col = "gray") +
  # 상자그림 : 투명도 50%
  geom_boxplot(alpha = .5) +
  ggtitle("Boxplot of passengers Fare") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15))

# multiplot layout 형식 지정
multi.layout = matrix(c(1, 1, 2, 2), 2, 2)

# 위에서 생성한 2개의 그래프 한 화면에 출력
multiplot(Fare.p1, Fare.p2, layout = multi.layout)

```



생존자들이 사망한 승객들보다 Fare가 더 높지만 그렇게 큰 차이는 없음을 알 수 있습니다.

Sex

남, 여 간에 생존율의 차이가 있을까요? 아래의 plot들을 보시기 바랍니다.

```
sex.p1 <- full %>%
  dplyr::group_by(Sex) %>%
  summarize(N = n()) %>%
  ggplot(aes(Sex, N)) +
  geom_col() +
  geom_text(aes(label = N), size = 5, vjust = 1.2, color = "#FFFFFF") +
  ggtitle("Bar plot of Sex") +
  labs(x = "Sex", y = "Count")

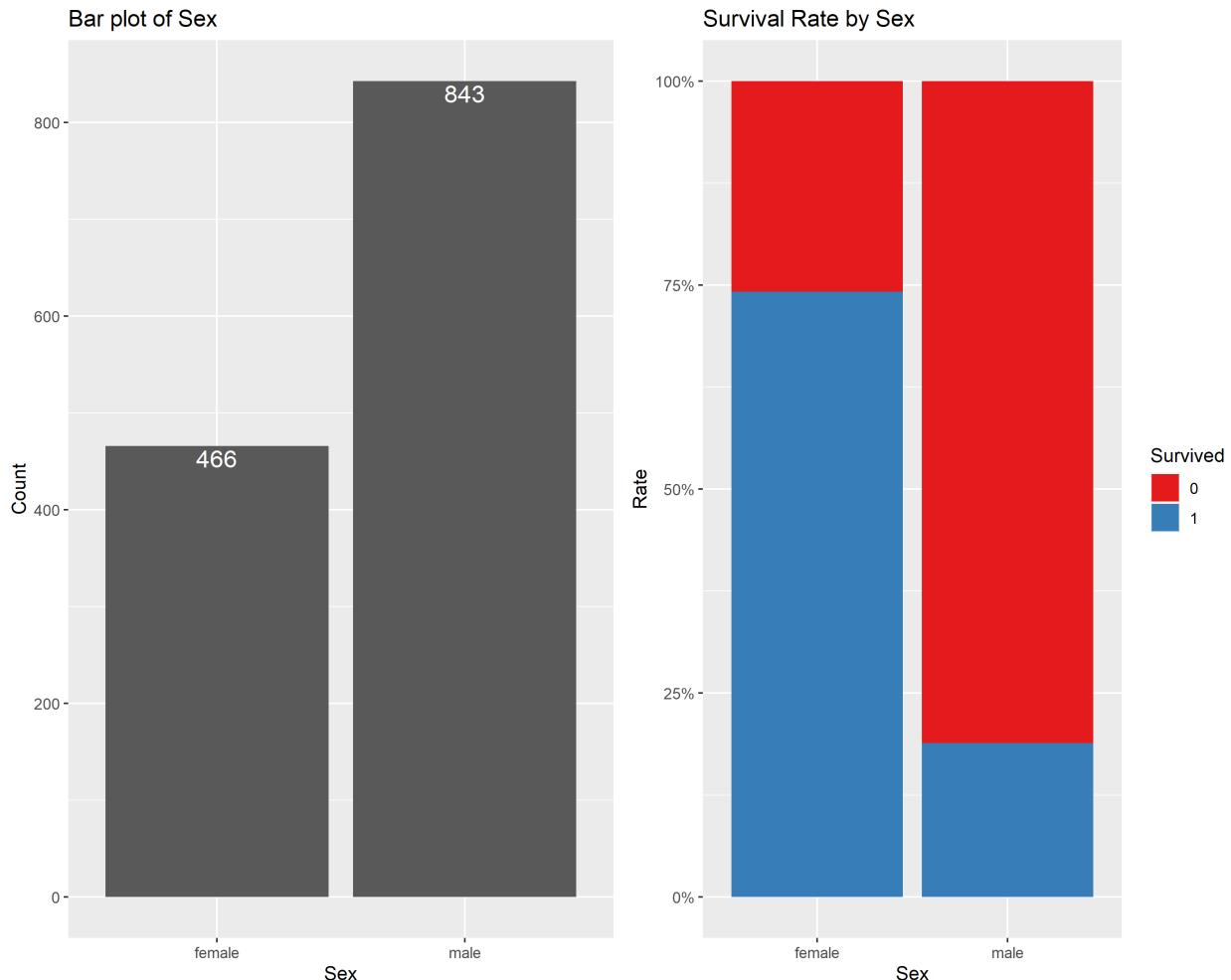
sex.p2 <- full[1:891, ] %>%
  ggplot(aes(Sex, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(labels = percent) +
  ggtitle("Survival Rate by Sex") +
  labs(x = "Sex", y = "Rate")
```

```

multi.layout = matrix(rep(c(1, 2), times = 2), 2, 2, byrow = T)

multiplot(sex.p1, sex.p2, layout = multi.layout)

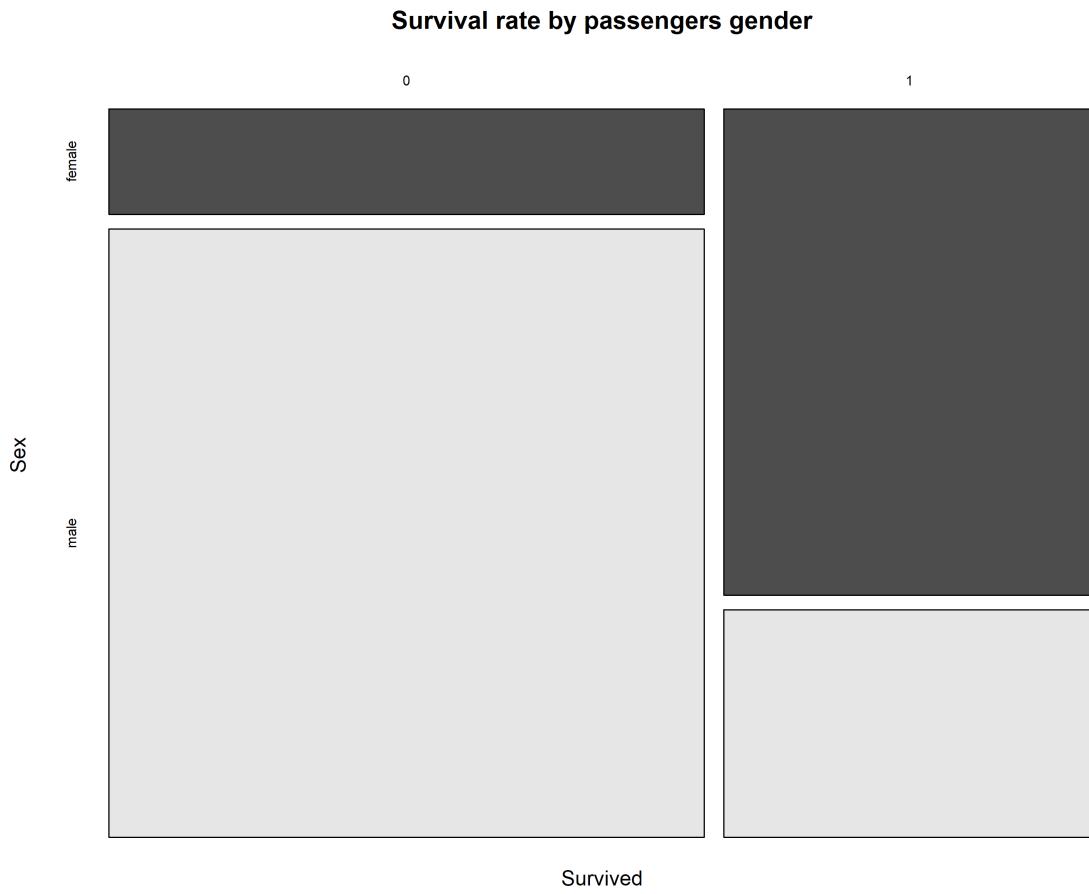
```



```

mosaicplot(Survived ~ Sex,
           data = full[1:891, ], col = TRUE,
           main = "Survival rate by passengers gender")

```



그래프를 해석해보면, 남성이 여성보다 훨씬 많은 반면에 생존율은 여성 탑승객이 높음을 알 수 있습니다.

데이터 전처리 및 특징 확인

Age -> Age.Group

```
full <- full %>%
  # 결측치(NA)를 먼저 채우는데 결측치를 제외한 값들의 평균으로 채움.
  mutate(Age = ifelse(is.na(Age), mean(full$Age, na.rm = TRUE), Age),
        # Age 값에 따라 범주형 파생 변수 Age.Group 를 생성
        Age.Group = case_when(Age < 13 ~ "Age.0012",
                               Age >= 13 & Age < 18 ~ "Age.1317",
                               Age >= 18 & Age < 60 ~ "Age.1859",
                               Age >= 60 ~ "Age.60inf"),
        # Chr 속성을 Factor로 변환
        Age.Group = factor(Age.Group))
```

SibSp & Parch -> FamilySized

```
full <- full %>%
  # SibSp, Parch와 1(본인)을 더해서 FamilySize라는 파생변수를 먼저 생성
```

```

mutate(FamilySize = .\$SibSp + .\$Parch + 1,
       # FamilySize 의 값에 따라서 범주형 파생 변수 FamilySized 를 생성
       FamilySized = dplyr::case_when(FamilySize == 1 ~ "Single",
                                         FamilySize >= 2 & FamilySize < 5 ~ "Small",
                                         FamilySize >= 5 ~ "Big"),
       # Chr 속성인 FamilySized를 factor로 변환하고
       # 집단 규모 크기에 따라 levels를 새로 지정
       FamilySized = factor(FamilySized, levels = c("Single", "Small", "Big")))

```

SibSp, Parch를 이용해서 FamilySized를 만들었습니다. 이렇게 두 개의 변수를 하나로 줄이면 모델이 더욱 단순해지는 장점이 있습니다. 비슷한 사용 방법으로는 키와 체중을 합쳐서 BMI 지수로 만드는 것 입니다.

2.4 Name & Sex -> title

앞의 결과를 봤을 때 여성의 생존율이 남성보다 높은 것을 확인했습니다. 따라서 Name에서 “성별과 관련된 이름만을 추출해서 범주화 시키면 쓸모있지 않을까?”라는 생각이듭니다. 먼저 full data에서 Name이라는 열벡터만 추출 한 뒤에 title로 저장합니다.

```

# 우선 Name 열벡터만 추출해서 title 벡터에 저장
title <- full$Name

# 정규표현식과 gsub()을 이용해서 성별과 관련성이 높은 이름만 추출해서 title 벡터로 저장
title <- gsub("^(.*?)([Mm]r|[Ff]r|[Dd]r|[Ll]ady|[Cc]apt|[Oo]fficer|[Ss]ir|[Mm]iss|[Mm]aster|[Dd]on|[Mm]ajor|[Cc]ountess|[Jj]onkheer)$", "\\1", title)

# 위에서 저장한 title 벡터를 다시 full 에 저장하되, title 파생변수로 저장
full$title <- title

```

그 다음 고유한(Unique한) title들에는 어떤 것들이 있는지 확인해봅니다.

```
unique(full$title)
```

```

## [1] "Mr"          "Mrs"         "Miss"        "Master"       "Don"
## [6] "Rev"         "Dr"          "Mme"         "Ms"          "Major"
## [11] "Lady"        "Sir"          "Mlle"        "Col"         "Capt"
## [16] "the Countess" "Jonkheer"    "Dona"

```

총 18개의 범주가 있음을 알 수 있습니다. 이 title이라는 파생변수를 그대로 사용할 경우 모델의(특히 Tree based model) 복잡도가 상당히 높아지기 때문에 범주를 줄여줘야합니다. 그 전에 descr패키지를 이용해서 각 범주별 빈도수와 비율을 확인해보겠습니다.

```
# 범주별 빈도수, 비율 확인
descr::CrossTable(full$title)
```

```

##      Cell Contents
## |-----|
## |                               N |
## |           N / Row Total |
## |-----|
## 
## |      Capt |       Col |       Don |      Dona |       Dr |
## |-----|-----|-----|-----|-----|
## |      1 |       4 |       1 |      1 |       8 |
## | 0.001 | 0.003 | 0.001 | 0.001 | 0.006 |
## |-----|-----|-----|-----|-----|
## 
## |      Jonkheer |       Lady |      Major |      Master |      Miss |
## |-----|-----|-----|-----|-----|

```

```

## | 1 | 1 | 2 | 61 | 260 |
## | 0.001 | 0.001 | 0.002 | 0.047 | 0.199 |
## |-----|-----|-----|-----|-----|
## | Mlle | Mme | Mr | Mrs | Ms |
## |-----|-----|-----|-----|-----|
## | 2 | 1 | 757 | 197 | 2 |
## | 0.002 | 0.001 | 0.578 | 0.150 | 0.002 |
## |-----|-----|-----|-----|-----|
## | Rev | Sir | the Countess |
## |-----|-----|-----|
## | 8 | 1 | 1 |
## | 0.006 | 0.001 | 0.001 |
## |-----|-----|-----|

```

18개나 되는 범주들의 빈도수와 비율이 너무 제각각입니다. 따라서 이것들을 총 5개 범주로 줄이도록 하겠습니다.

5개 범주로 단순화 시키는 작업

```

full <- full %>%
  # "%in%" 대신 "=="을 사용하게되면 Recycling Rule 때문에 원하는대로 되지 않습니다.
  mutate(title = ifelse(title %in% c("Mlle", "Ms", "Lady", "Dona"), "Miss", title),
         title = ifelse(title == "Mme", "Mrs", title),
         title = ifelse(title %in% c("Capt", "Col", "Major", "Dr", "Rev", "Don",
                                      "Sir", "the Countess", "Jonkheer"), "Officer", title),
         title = factor(title))

```

파생변수 생성 후 각 범주별 빈도수, 비율 확인

```
descr::CrossTable(full$title)
```

```

## Cell Contents
## |-----|
## | N |
## | N / Row Total |
## |-----|
## | Master | Miss | Mr | Mrs | Officer |
## |-----|-----|-----|-----|-----|
## | 61 | 266 | 757 | 198 | 27 |
## | 0.047 | 0.203 | 0.578 | 0.151 | 0.021 |
## |-----|-----|-----|-----|-----|

```

Ticket -> ticket.size

승객은 (train, test 합쳐서 모두) 1309명입니다. 그런데 모든 승객의 티켓이 다 다르지 않습니다. 아래 summary()와 unique()의 결과를 보시기 바랍니다.

고유한(unique한) 범주의 갯수만 파악하려고 length()를 사용했습니다.

```
length(unique(full$Ticket))
```

```
## [1] 929
```

모두 출력하면 너무 지저분해서 10개만 출력했습니다.

```
head(summary(full$Ticket), 10)
```

	CA.	2343	1601	CA	2144	3101295	347077	347082
##		11		8		7	7	7

```

##      PC 17608 S.O.C. 14879      113781      19950
##      7           7           6           6

```

결측치가 없는 feature인데 왜 고유한 티켓이 929개 일까요? 심지어 티켓이 CA. 2343으로 완전히 같은 인원이 11명이나 됩니다. 어떤 승객들인지 확인해보겠습니다.

```

full %>%
  # 티켓이 일치하는 11명의 승객들만 필터링
  filter(Ticket == "CA. 2343") %>%
  # 모든 변수에 대해 확인할 필요는 없으므로 아래 변수들만 보려고 합니다.
  select(Pclass, Name, Age, FamilySized)

```

```

## # A tibble: 11 x 4
##   Pclass Name                               Age FamilySized
##   <dbl> <fct>                           <dbl> <fct>
## 1 3    "Sage, Master. Thomas Henry"     29.9 Big
## 2 3    "Sage, Miss. Constance Gladys"  29.9 Big
## 3 3    "Sage, Mr. Frederick"          29.9 Big
## 4 3    "Sage, Mr. George John Jr"     29.9 Big
## 5 3    "Sage, Miss. Stella Anna"       29.9 Big
## 6 3    "Sage, Mr. Douglas Bullen"      29.9 Big
## 7 3    "Sage, Miss. Dorothy Edith \"Dolly\"" 29.9 Big
## 8 3    "Sage, Miss. Ada"                29.9 Big
## 9 3    "Sage, Mr. John George"         29.9 Big
## 10 3   "Sage, Master. William Henry"   14.5 Big
## 11 3   "Sage, Mrs. John (Annie Bullen)" 29.9 Big

```

위 11명의 승객들은 모두 같은 가족, 형제들인 것을 알 수 있습니다. 이렇게 티켓이 완전히 동일한 승객들이 있는 반면에 일부분만 일치하는 승객들도 존재합니다. 이런 티켓의 고유한 넘버(글자수) 갯수를 나타내는 ticket.unique 파생변수를 만들고 ticket.unique를 바탕으로 3개 범주를 갖는 ticket.size 파생변수를 만들어 봅시다.

```

# 우선 ticket.unique가 모두 0이라고 저장함
ticket.unique <- rep(0, nrow(full))

# Ticket Feature에서 고유한 것들만 추출해서 tickets 벡터에 저장
tickets <- unique(full$Ticket)

# 반복문을 중첩 활용해서 티켓이 같은 승객들만 추출 후, 각 티켓들의 길이(문자 갯수)를 추출해서 저장한다.
for (i in 1:length(tickets)) {
  current.ticket <- tickets[i]
  party.indexes <- which(full$Ticket == current.ticket)
  # For loop 중첩
  for (k in 1:length(party.indexes)) {
    ticket.unique[party.indexes[k]] <- length(party.indexes)
  }
}

# 위에서 계산한 ticket.unique 을 파생변수로 저장
full$ticket.unique <- ticket.unique

# ticket.unique에 따라 세가지 범주로 나눠서 ticket.size 변수 생성
full <- full %>%
  mutate(ticket.size = case_when(ticket.unique == 1 ~ 'Single',
                                 ticket.unique < 5 & ticket.unique >= 2 ~ "Small",
                                 ticket.unique >= 5 ~ "Big"),

```

```

ticket.size = factor(ticket.size,
                      levels = c("Single", "Small", "Big")))

```

Embarked

결측치(Missing value, NA)가 2개 있던 feature입니다. Embarked의 경우 3개 범주 중에서 가장 최빈값인 S로 치환하도록 합니다.

```

full$Embarked <- replace(full$Embarked,
                           which(is.na(full$Embarked)), # 치환할 Data$feature 지정
                           'S')                         # 결측치들만 찾기
                                         # 치환할 값 지정

```

Fare

Fare의 경우 결측치가 유일하게 1개 있었습니다. 위에서 본 히스토그램(Chapter 3.5 Fare)을 바탕으로 결측치를 0으로 치환해줍니다.

```
full$Fare <- replace(full$Fare, which(is.na(full$Fare)), 0)
```

여기까지 하시면 데이터 전처리가 모두 끝난 것입니다. 다음은 지금까지 만든 파생변수들을 탐색하면서 모델 생성에 사용할 변수들을 선택하는 과정입니다. 즉, Feature selection이라고 보시면 됩니다.

Survived와 관련된 특징 선택

본격적인 시각화에 앞서, 여기서는 각 변수들이 생존율과 얼마나 연관성이 높은지를 보는것이 목적이기 때문에 full data 전체를 사용하지 않고, 생존과 사망여부를 알 수 있는 train data set만 사용하였습니다. 그리고 위에서 사용한 plot들이 그대로 중복된 경우도 있으니 참고하시기 바랍니다.

데이터셋 분리

먼저 아래 코드를 이용해서 전처리가 끝난 full data를 train, test로 분할합니다.

```

# feature selection 전이므로 우선은 모든 변수들을 선택합니다.
train <- full[1:891, ]
test <- full[892:1309, ]

```

Pclass

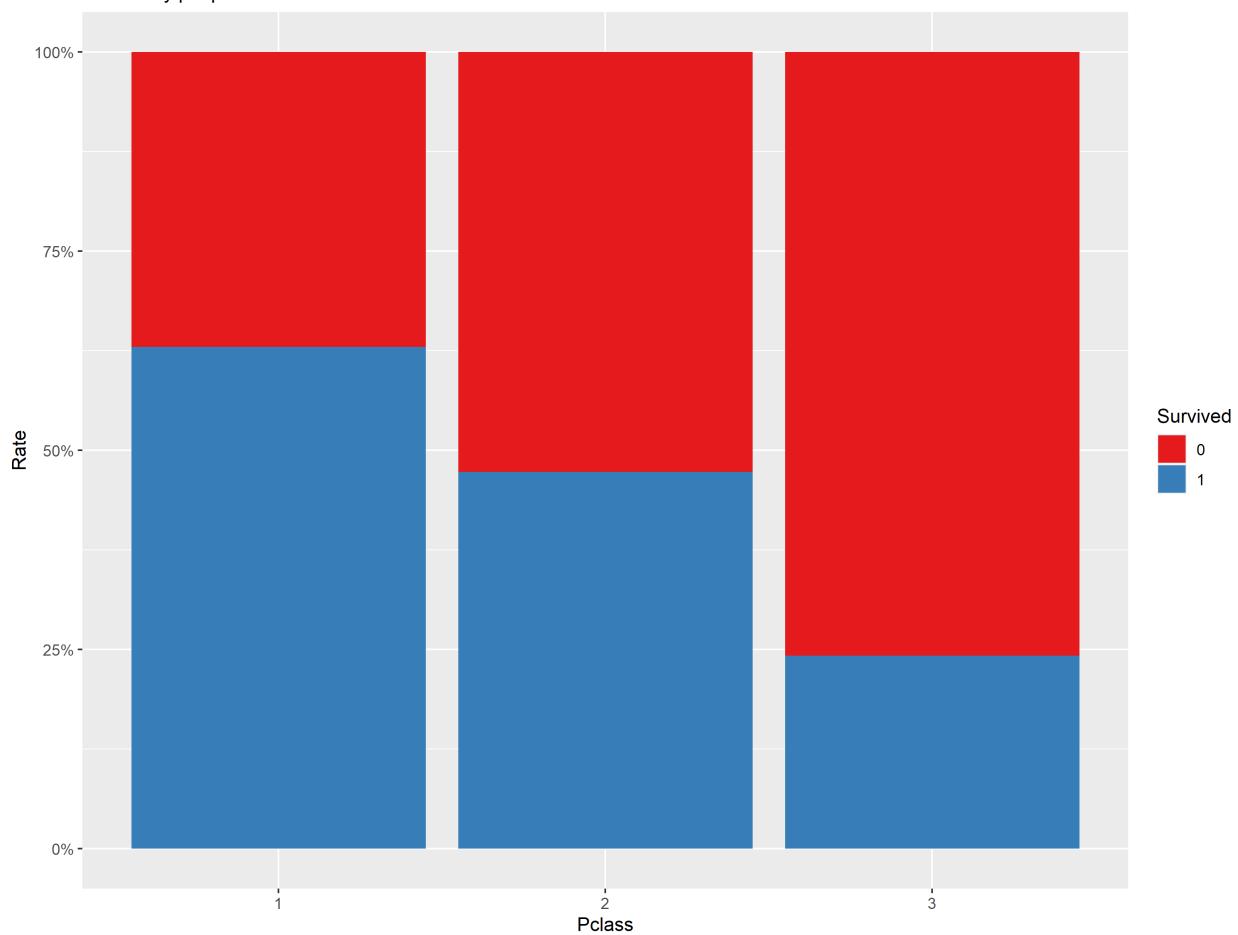
```

train %>%
  ggplot(aes(Pclass, fill = Survived)) +
  geom_bar(position = "fill") +
  # plot 테마 설정 : 조금 더 선명한 색깔로 변환해준다.
  scale_fill_brewer(palette = "Set1") +
  # Y axis setting
  scale_y_continuous(labels = percent) +
  # x, y 축 이름과 plot의 main title, sub title 설정
  labs(x = "Pclass", y = "Rate",
       title = "Bar plot", subtitle = "How many people survived in each Pclass?")

```

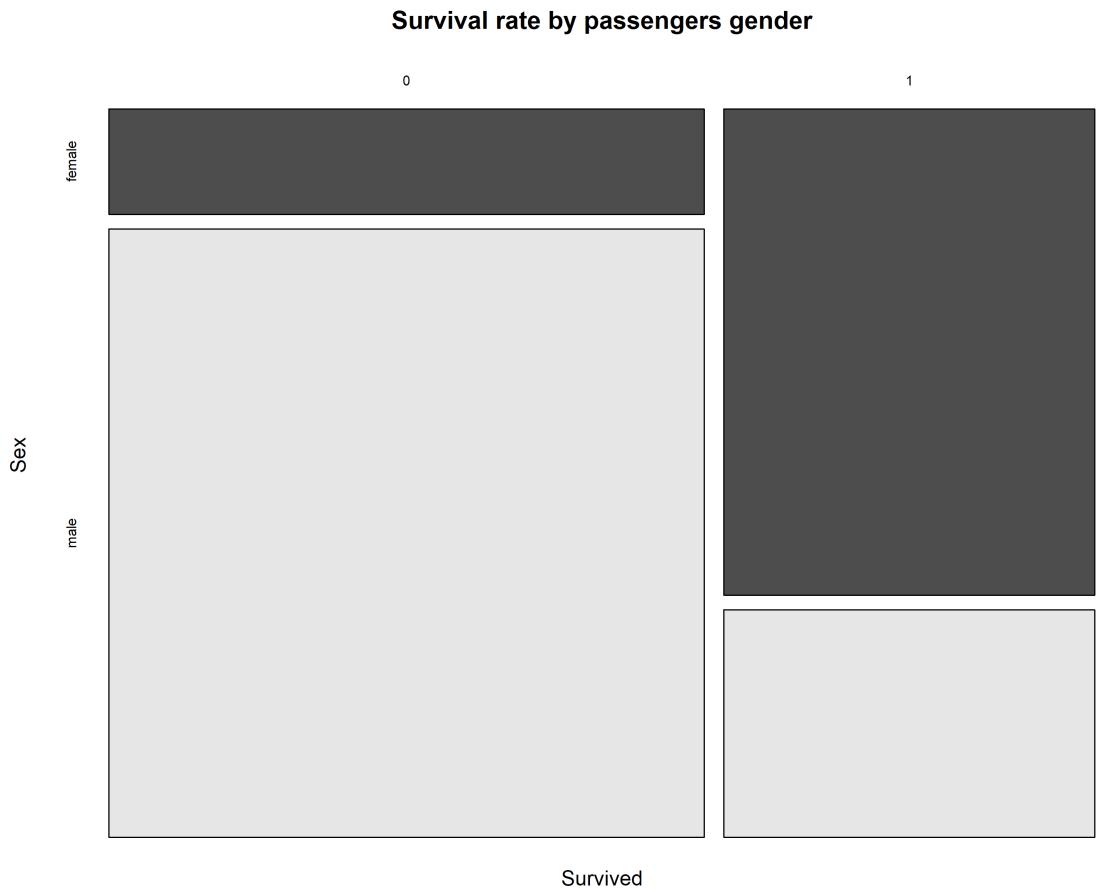
Bar plot

How many people survived in each Pclass?



Sex

```
mosaicplot(Survived ~ Sex,
            data = train, col = TRUE,
            main = "Survival rate by passengers gender")
```

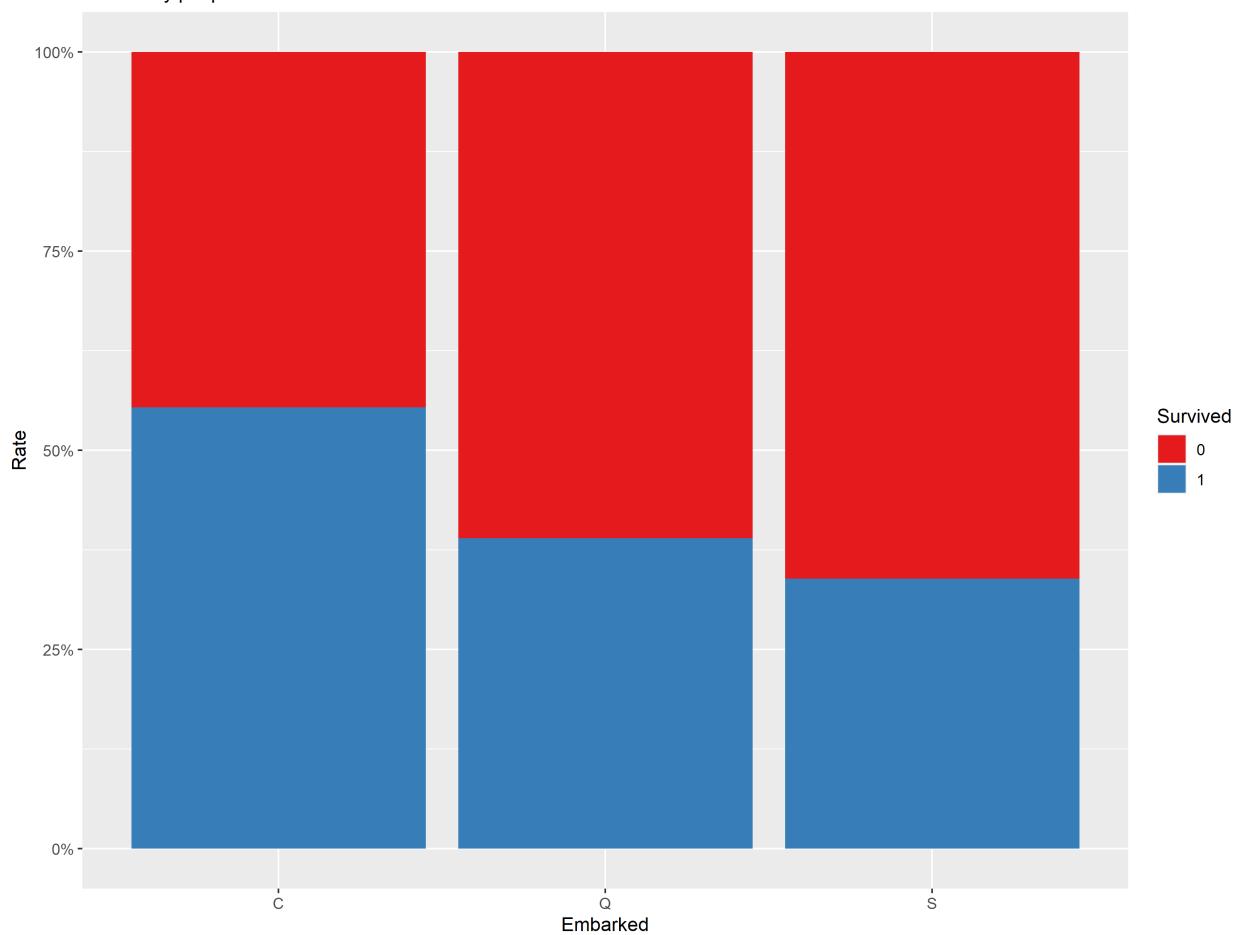


Embarked

```
train %>%
  ggplot(aes(Embarked, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(labels = percent) +
  labs(x = "Embarked", y = "Rate",
       title = "Bar plot", subtitle = "How many people survived in each Embarked?")
```

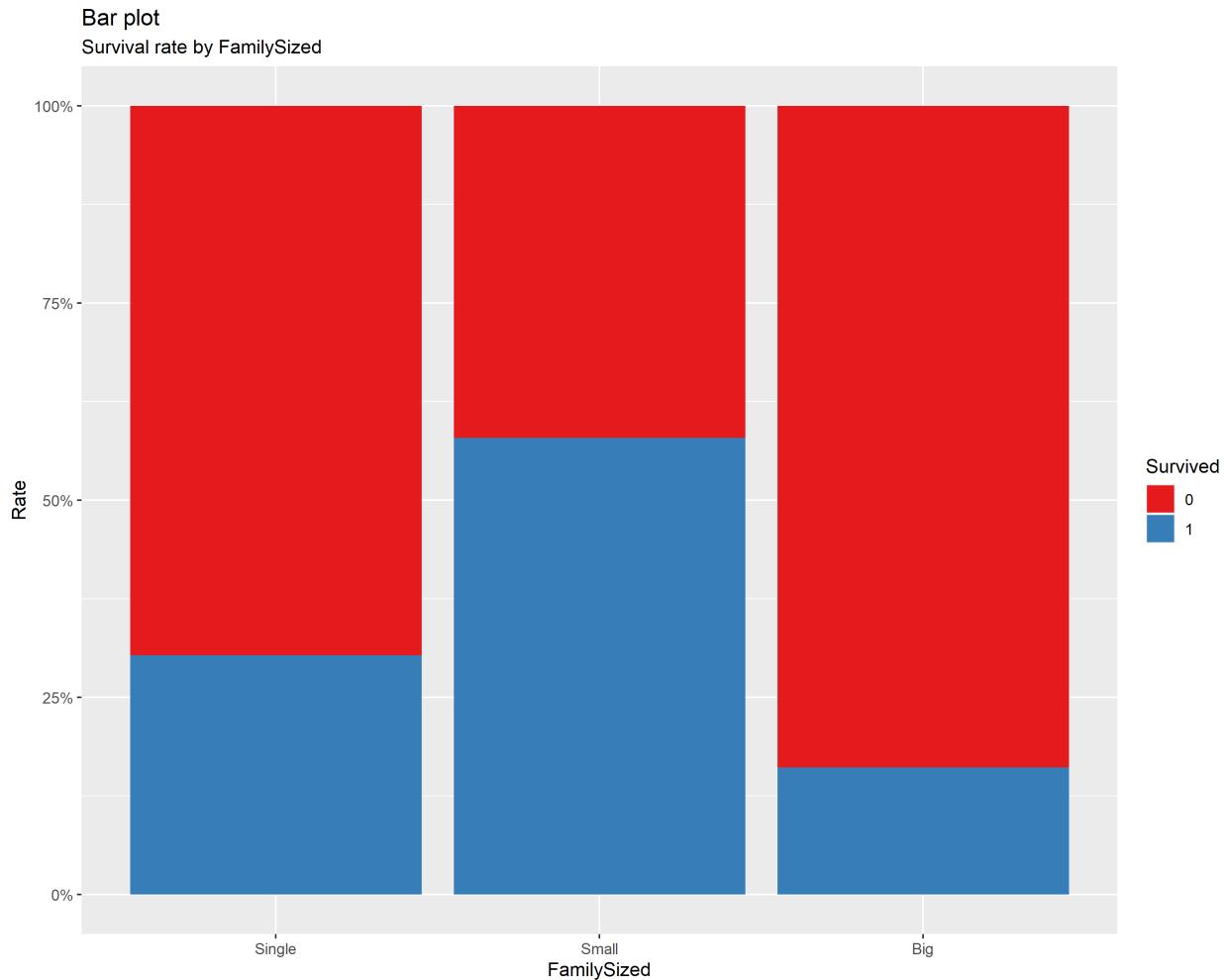
Bar plot

How many people survived in each Embarked?



FamilySized

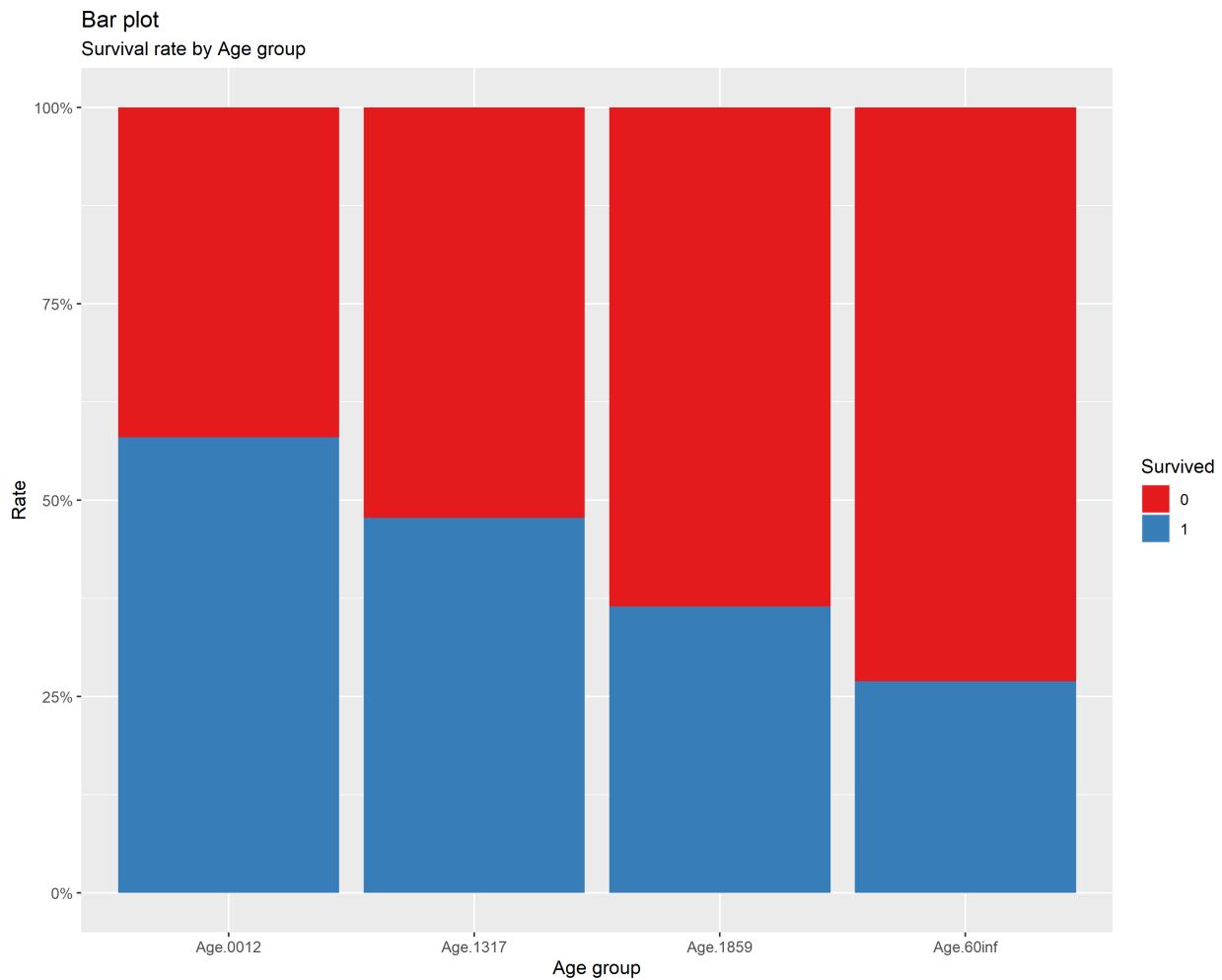
```
train %>%
  ggplot(aes(FamilySized, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(labels = percent) +
  labs(x = "FamilySized", y = "Rate",
       title = "Bar plot", subtitle = "Survival rate by FamilySized")
```



동승한 인원수에 따라 생존율에 차이가 있고 FamilySized와 Survived는 비선형 관계임을 알 수 있습니다.

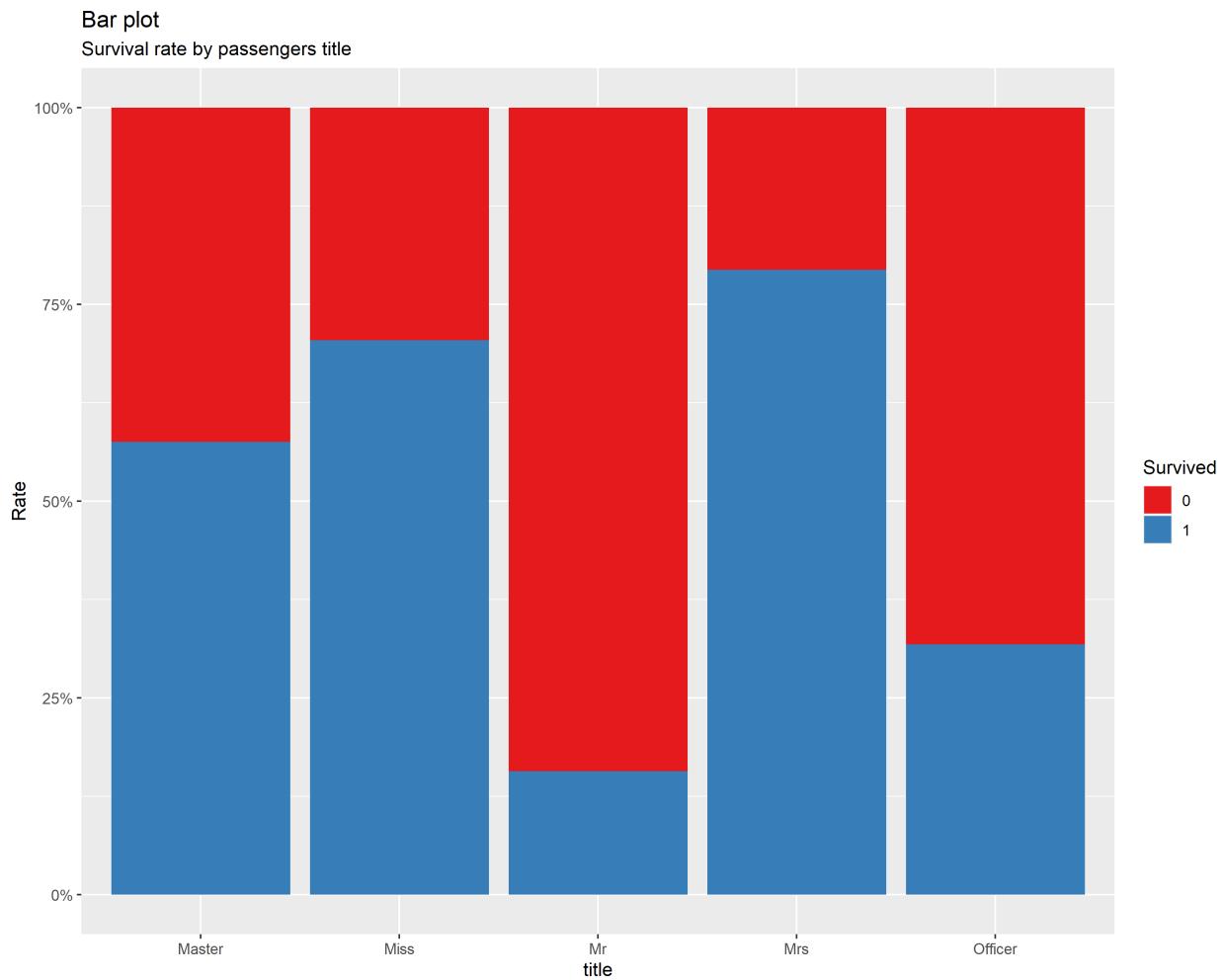
Age.Group

```
train %>%
  ggplot(aes(Age.Group, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(labels = percent) +
  labs(x = "Age group", y = "Rate",
       title = "Bar plot", subtitle = "Survival rate by Age group")
```



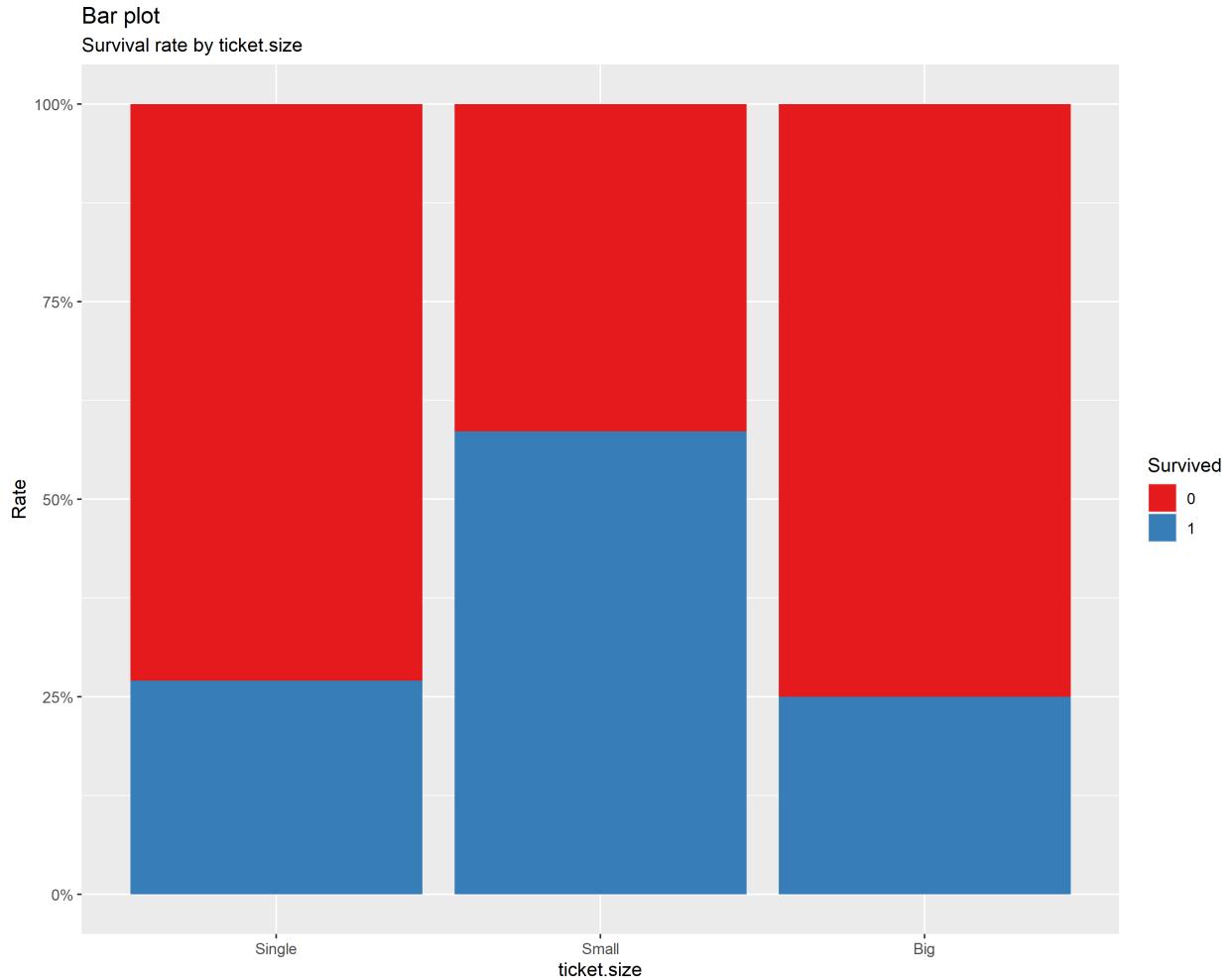
title

```
train %>%
  ggplot(aes(title, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(labels = percent) +
  labs(x = "title", y = "Rate",
       title = "Bar plot", subtitle = "Survival rate by passengers title")
```



`ticket.size`

```
train %>%
  ggplot(aes(ticket.size, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Set1") +
  scale_y_continuous(labels = percent) +
  labs(x = "ticket.size", y = "Rate",
       title = "Bar plot", subtitle = "Survival rate by ticket.size")
```



특징 추출

지금까지 생성한 파생변수들이 모두 유용함을 알았으니 실제로 사용할 변수들만 선택해서 저장하도록 합니다. 아래 표는 실제 선택한 변수들에 대한 간단한 설명들입니다.

변수명	Type	설명
Survived	factor	Target feature, 생존 == 1, 사망 == 0
Sex	factor	성별, male or female
Pclass	factor	선실 등급, 1등급(1), 2등급(2), 3등급(3)
Embarked	factor	승선항, 사우샘프턴(S), 셀부르(C), 퀸즈타운(Q)
FamilySized	factor	가족의 규모, SibSp와 Parch를 이용해서 만든 파생변수, 범주는 3개
Age.Group	factor	연령대, Age를 이용해서 만든 파생변수, 범주는 4개
title	factor	이름의 일부분, Name을 이용해서 만든 파생변수, 범주는 5개
ticket.size	factor	티켓의 고유한 부분의 길이, ticket을 이용해서 만든 파생변수, 범주는 3개

```
# Id number 제외하고 실제로 사용할 7개 입력변수와 1개의 타겟변수를 선택, 저장
train <- train %>%
  select("Pclass", "Sex", "Embarked", "FamilySized",
         "Age.Group", "title", "ticket.size", "Survived")
```

```
# Submit을 위해서 Id 열벡터 추출해서 ID에 저장
ID <- test$PassengerId

# Id와 Survived를 제외한 나머지 6개 변수들을 선택, 저장
test <- test %>%
  select("Pclass", "Sex", "Embarked", "FamilySized",
  "Age.Group", "title", "ticket.size")
```

머신러닝 모델

본격적으로 train data set을 이용해서 기계학습 모델을 생성할 차례입니다. 원래는 `train`, `validation`, `test` data set들을 먼저 만들고 다양한 모델들을 생성 한 후에 교차검증(CV, Cross Validation)를 거쳐서 최종 모델을 선택하는게 맞지만 여기서는 그런 과정들을 생략하고 `RandomForest` 만을 생성한 뒤에 `test` data를 예측(추정)해보고 competition에 Submit할 data를 만드는 것 까지 해보겠습니다.

랜덤포레스트

```
# 재현성을 위해서 seed number를 설정해줍니다.
set.seed(42)

titanic.rf <- randomForest(Survived ~ ., data = train, importance = T, ntree = 2000)
```

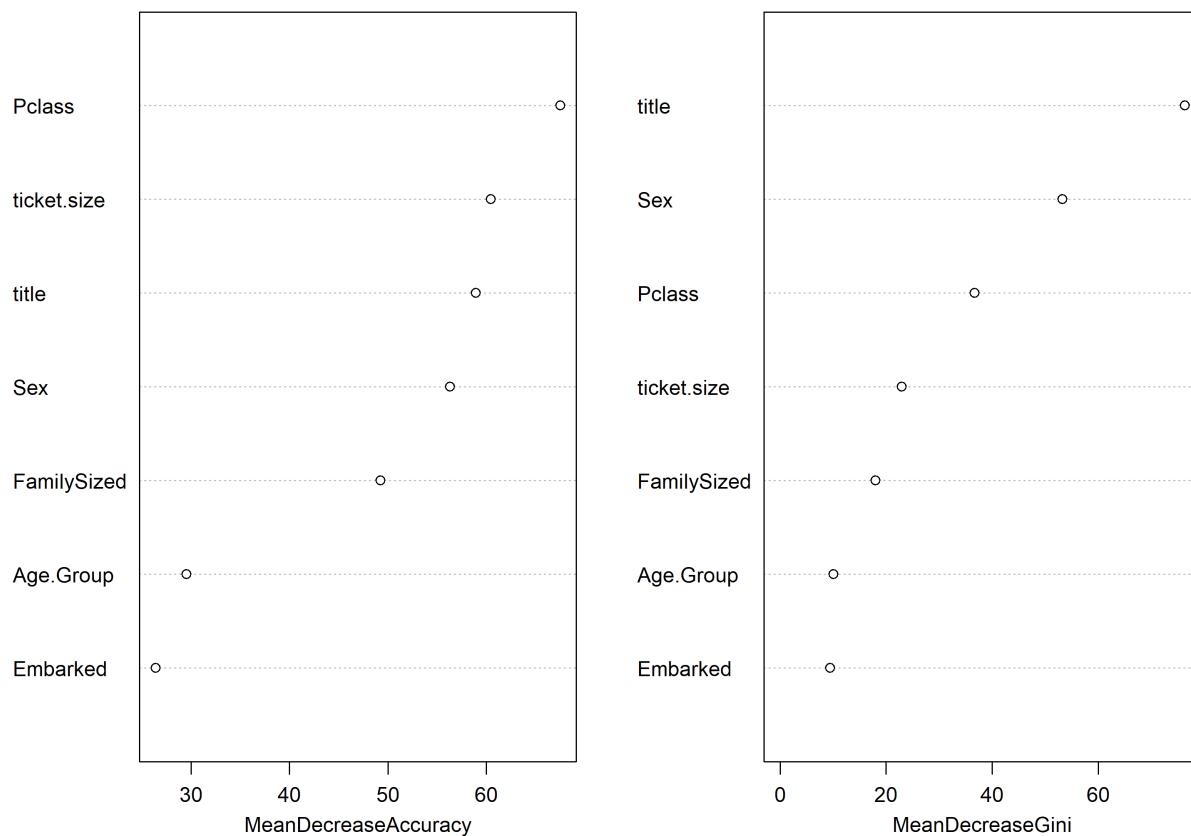
Feature importance check

```
importance(titanic.rf)

##          0      1 MeanDecreaseAccuracy MeanDecreaseGini
## Pclass    47.86179 56.15749       67.48707     36.660540
## Sex      52.49325 35.40070       56.31190     53.237389
## Embarked -7.40187 36.62916       26.40563      9.456708
## FamilySized 30.05571 30.52024       49.22725    18.030200
## Age.Group 14.48197 28.09305       29.56499    10.108311
## title    50.48199 43.29531       58.91748    76.306255
## ticket.size 41.63013 37.44123       60.44619    22.916008

varImpPlot(titanic.rf)
```

titanic.rf



예측 및 submit 파일 작성

```
# Prediction
pred.rf <- predict(object = titanic.rf, newdata = test, type = "class")

# Data frame generation
submit <- data.frame(PassengerID = ID, Survived = pred.rf)

# Write the submit data frame to file : setwd()로 지정해놓은 폴더에 csv로 생성됩니다.
write.csv(submit, file = './titanic_submit.csv', row.names = F)
```