```r
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────────────────────────────────────────── tidyver
se 1.2.1 ──
```

```
## ✔ ggplot2 3.0.0     ✔ purrr   0.2.4
## ✔ tibble  1.4.2     ✔ dplyr   0.7.6
## ✔ tidyr   0.8.0     ✔ stringr 1.3.1
## ✔ readr   1.1.1     ✔ forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

```
## ── Conflicts ───────────────────────────────────────────────────────────────────────────── tidyverse_con
flicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```r
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 3.4.4
```

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.4.4
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
library(rworldmap)
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 3.4.4
```

```
## ### Welcome to rworldmap ###
```

```
## For a short introduction type :   vignette('rworldmap')
```

```r
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(knitr)
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.4.4
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(RCurl)
```

```
## Warning: package 'RCurl' was built under R version 3.4.4
```

```
## Loading required package: bitops
```

```
##
## Attaching package: 'RCurl'
```

```
## The following object is masked from 'package:tidyr':
##
##     complete
```

```r
library(leaps)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loaded glmnet 2.0-16
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

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

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'
```

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

```r
library(ROCR)
library(mlbench)
library(caret)
library(caretEnsemble)
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.4.4
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(PCAmixdata)
```

```r
ksp <- read.csv("~/Downloads/ks-projects-201801.csv")
```

1 Data Cleaning

```
sum(is.na(ksp))
```

```
## [1] 3797
```

```
str(ksp)
```

```
## 'data.frame':    378661 obs. of  15 variables:
##  $ ID              : int  1000002330 1000003930 1000004038 1000007540 1000011046 1000014025 1000023410 1
000030581 1000034518 100004195 ...
##  $ name            : Factor w/ 375765 levels "","    IT'S A HOT CAPPUCCINO NIGHT  ",..: 332493 135633 36
4946 344770 77274 206067 293430 69281 284103 290686 ...
##  $ category        : Factor w/ 159 levels "3D Printing",..: 109 94 94 91 56 124 59 42 114 40 ...
##  $ main_category   : Factor w/ 15 levels "Art","Comics",..: 13 7 7 11 7 8 8 8 5 7 ...
##  $ currency        : Factor w/ 14 levels "AUD","CAD","CHF",..: 6 14 14 14 14 14 14 14 14 14 ...
##  $ deadline        : Factor w/ 3164 levels "2009-05-03","2009-05-16",..: 2288 3042 1333 1017 2247 2463 19
96 2448 1790 1863 ...
##  $ goal            : num  1000 30000 45000 5000 19500 50000 1000 25000 125000 65000 ...
##  $ launched        : Factor w/ 378089 levels "1970-01-01 01:00:00",..: 243292 361975 80409 46557 235943
278600 187500 274014 139367 153766 ...
##  $ pledged         : num  0 2421 220 1 1283 ...
##  $ state           : Factor w/ 6 levels "canceled","failed",..: 2 2 2 2 1 4 4 2 1 1 ...
##  $ backers         : int  0 15 3 1 14 224 16 40 58 43 ...
##  $ country         : Factor w/ 23 levels "AT","AU","BE",..: 10 23 23 23 23 23 23 23 23 23 ...
##  $ usd.pledged     : num  0 100 220 1 1283 ...
##  $ usd_pledged_real: num  0 2421 220 1 1283 ...
##  $ usd_goal_real   : num  1534 30000 45000 5000 19500 ...
```

```
sapply(ksp, function(x) sum(is.na(x)))
```

```
##               ID             name         category    main_category
##                0                0                0                0
##         currency         deadline             goal         launched
##                0                0                0                0
##          pledged            state          backers          country
##                0                0                0                0
##      usd.pledged usd_pledged_real    usd_goal_real
##             3797                0                0
```

```
sapply(ksp, function(x) sum(is.null(x)))
```

```
##               ID             name         category    main_category
##                0                0                0                0
##         currency         deadline             goal         launched
##                0                0                0                0
##          pledged            state          backers          country
##                0                0                0                0
##      usd.pledged usd_pledged_real    usd_goal_real
##                0                0                0
```

```
#usd.pledged has 3797 missing values. I will just replace the value to the mean of its column.
```

```
ksp$usd.pledged <- ifelse(is.na(ksp$usd.pledged), mean(na.omit(ksp$usd.pledged)), ksp$usd.pledged)
sapply(ksp, function(x) sum(is.na(x)))
```

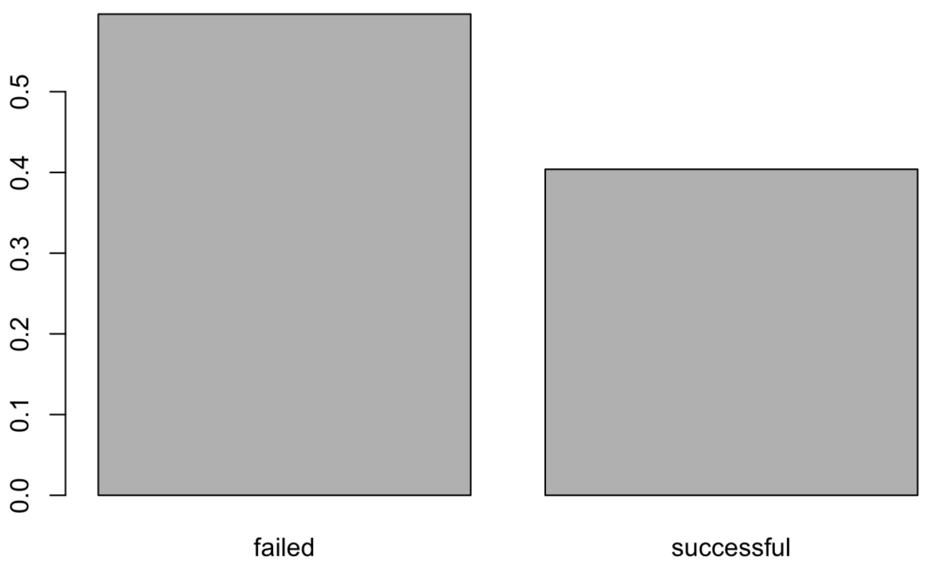
```
##               ID             name         category    main_category
##                0                0                0                0
##         currency         deadline             goal         launched
##                0                0                0                0
##          pledged            state          backers          country
##                0                0                0                0
##       usd.pledged usd_pledged_real    usd_goal_real
##                0                0                0
```

```
ksp$ID <- as.character(ksp$ID)
ksp$name <- as.character(ksp$name)

#Now I have no missing values in the dataset
```

```
ksp.new <- ksp[ksp$state == 'failed' | ksp$state == 'successful', ]
ksp.new$state <- as.character(ksp.new$state)
ksp.new$state <- as.factor(ksp.new$state)
prop.table(table(ksp.new$state))
```

```
##
##     failed successful
##  0.5961227  0.4038773
```

```
barplot(prop.table(table(ksp.new$state)))
```




```
#Since our target variable is state, I subsetted records that the state is either success or fail to make it
binary problem
#Success rate has been incresed to 40% (35% before) after dropping other states.
```

```
ksp.new$duration <- as.Date(ksp.new$deadline) - as.Date(ksp.new$launched)
ksp.new$duration <- as.numeric(ksp.new$duration)
#added a new variable called duration to understand how many days spent for each project
```

```
ksp.new <- ksp.new %>%
  separate(col = "deadline", into = c("deadline_year", "deadline_month", "deadline_day"), sep = "-") %>%
  separate(col = "launched", into = c("launched_year", "launched_month", "launched_day"), sep = "-")
#broke down the date variables to year, month and day
```

```
str(ksp.new)
```

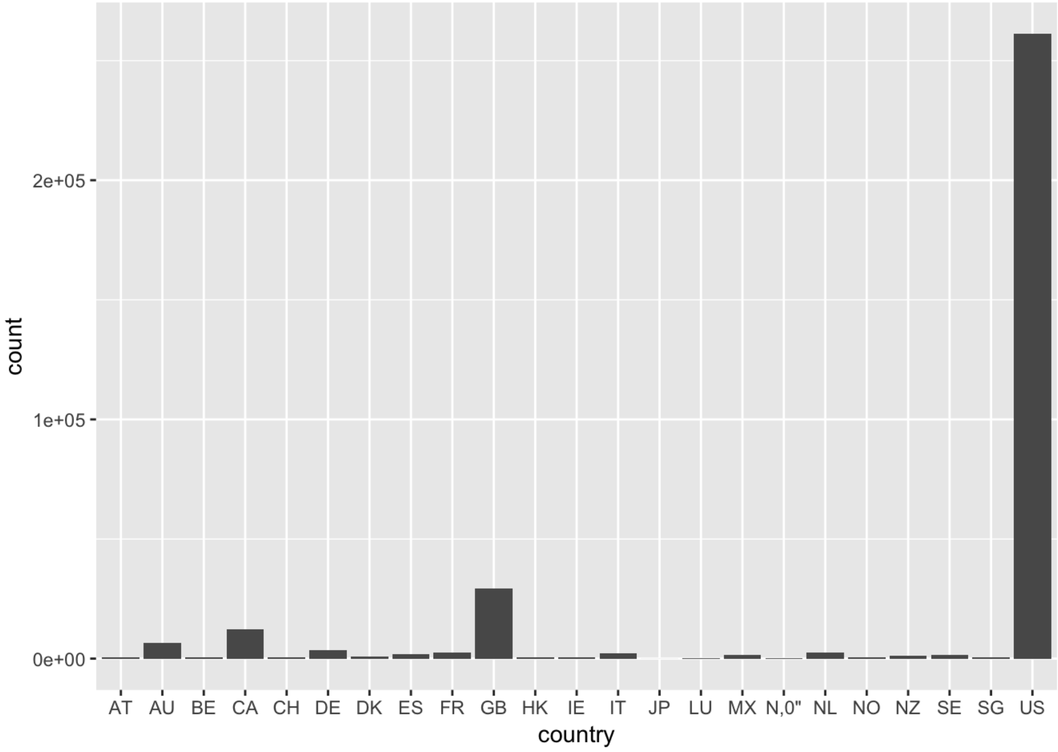
```
## 'data.frame':    331675 obs. of  20 variables:
##  $ ID              : chr  "1000002330" "1000003930" "1000004038" "1000007540" ...
##  $ name            : chr  "The Songs of Adelaide & Abullah" "Greeting From Earth: ZGAC Arts Capsule For E
T" "Where is Hank?" "ToshiCapital Rekordz Needs Help to Complete Album" ...
##  $ category        : Factor w/ 159 levels "3D Printing",..: 109 94 94 91 124 59 42 96 73 33 ...
##  $ main_category   : Factor w/ 15 levels "Art","Comics",..: 13 7 7 11 8 8 8 13 11 3 ...
##  $ currency        : Factor w/ 14 levels "AUD","CAD","CHF",..: 6 14 14 14 14 14 14 2 14 14 ...
##  $ deadline_year   : chr  "2015" "2017" "2013" "2012" ...
##  $ deadline_month  : chr  "10" "11" "02" "04" ...
##  $ deadline_day    : chr  "09" "01" "26" "16" ...
##  $ goal            : num  1000 30000 45000 5000 50000 1000 25000 2500 12500 5000 ...
##  $ launched_year   : chr  "2015" "2017" "2013" "2012" ...
##  $ launched_month  : chr  "08" "09" "01" "03" ...
##  $ launched_day    : chr  "11 12:12:28" "02 04:43:57" "12 00:20:50" "17 03:24:11" ...
##  $ pledged         : num  0 2421 220 1 52375 ...
##  $ state           : Factor w/ 2 levels "failed","successful": 1 1 1 1 2 2 1 1 2 1 ...
##  $ backers         : int  0 15 3 1 224 16 40 0 100 0 ...
##  $ country         : Factor w/ 23 levels "AT","AU","BE",..: 10 23 23 23 23 23 23 4 23 23 ...
##  $ usd.pledged     : num  0 100 220 1 52375 ...
##  $ usd_pledged_real: num  0 2421 220 1 52375 ...
##  $ usd_goal_real   : num  1534 30000 45000 5000 50000 ...
##  $ duration        : num  59 60 45 30 35 20 45 30 30 30 ...
```

```
ksp.new1 <- ksp.new[,c(1:4,6,7,10,11,5,16,15,18,19,20,14)]
str(ksp.new1)
```

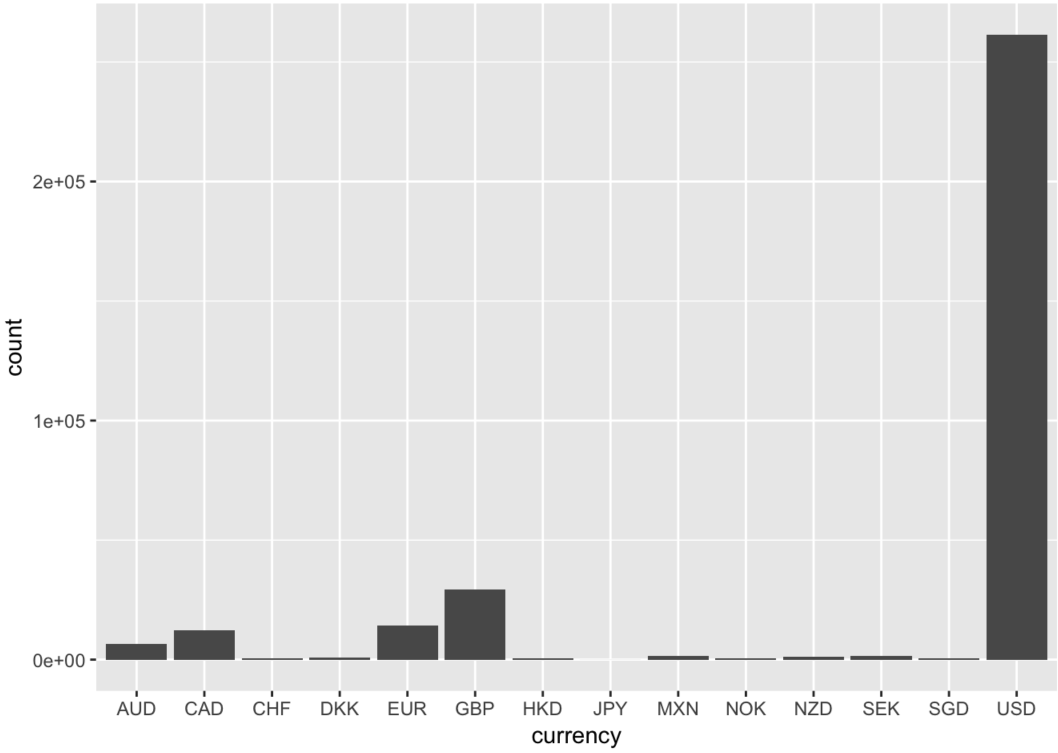
```
## 'data.frame':    331675 obs. of  15 variables:
##  $ ID              : chr  "1000002330" "1000003930" "1000004038" "1000007540" ...
##  $ name            : chr  "The Songs of Adelaide & Abullah" "Greeting From Earth: ZGAC Arts Capsule For E
T" "Where is Hank?" "ToshiCapital Rekordz Needs Help to Complete Album" ...
##  $ category        : Factor w/ 159 levels "3D Printing",..: 109 94 94 91 124 59 42 96 73 33 ...
##  $ main_category   : Factor w/ 15 levels "Art","Comics",..: 13 7 7 11 8 8 8 13 11 3 ...
##  $ deadline_year   : chr  "2015" "2017" "2013" "2012" ...
##  $ deadline_month  : chr  "10" "11" "02" "04" ...
##  $ launched_year   : chr  "2015" "2017" "2013" "2012" ...
##  $ launched_month  : chr  "08" "09" "01" "03" ...
##  $ currency        : Factor w/ 14 levels "AUD","CAD","CHF",..: 6 14 14 14 14 14 14 2 14 14 ...
##  $ country         : Factor w/ 23 levels "AT","AU","BE",..: 10 23 23 23 23 23 23 4 23 23 ...
##  $ backers         : int  0 15 3 1 224 16 40 0 100 0 ...
##  $ usd_pledged_real: num  0 2421 220 1 52375 ...
##  $ usd_goal_real   : num  1534 30000 45000 5000 50000 ...
##  $ duration        : num  59 60 45 30 35 20 45 30 30 30 ...
##  $ state           : Factor w/ 2 levels "failed","successful": 1 1 1 1 2 2 1 1 2 1 ...
```

```
#reordering columns
```

```
ggplot(ksp.new1, aes(country)) + geom_bar()
```




```
ggplot(ksp.new1, aes(currency)) + geom_bar()
```

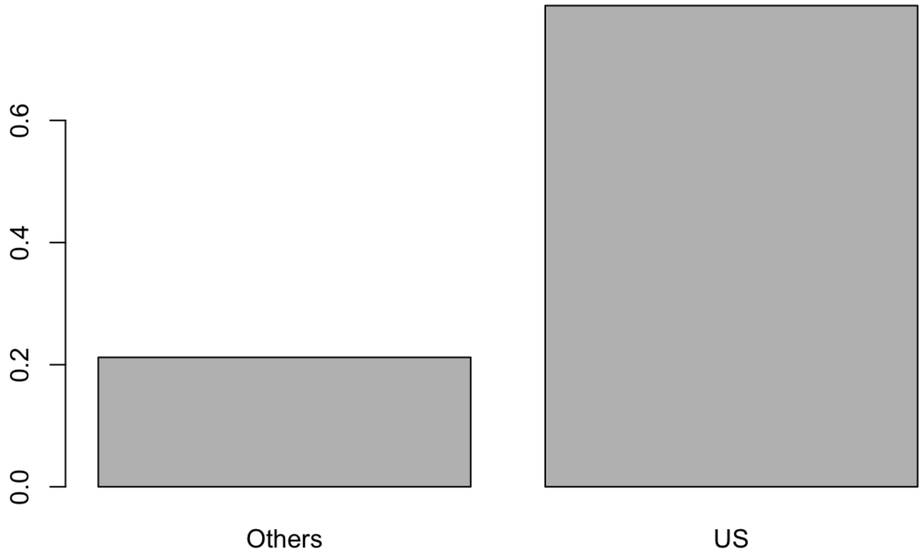



```
#when you see the graph below, most of the projects are took place in US. To reduce the level of columns, I'
m going to make it binary either us or not. Same for currency.
```

```
ksp.new1$country <- as.character(ksp.new1$country)
ksp.new1$country[ksp.new1$country %in% c("JP", "LU", "AT", "HK", "SG", "BE", "CH", "IE", "NO", "DK",
                                          "MX", "NZ", "SE", "ES", "IT", "NL", "FR", "DE","AU","CA", "GB",'N,0"
')] <- "Others"
ksp.new1$country <- as.factor(ksp.new1$country)
prop.table(table(ksp.new1$country))
```

```
##
##    Others        US
## 0.2119997 0.7880003
```

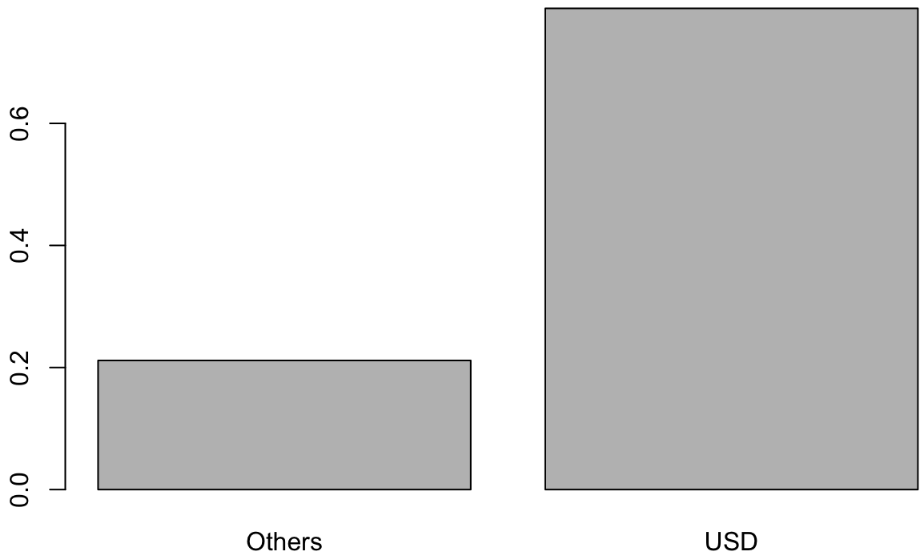
```
barplot(prop.table(table(ksp.new1$country)))
```




```
ksp.new1$currency <- as.character(ksp.new1$currency)
ksp.new1$currency[ksp.new1$currency %in% c("AUD","CHF","DKK","EUR","HKD","JPY","MXN","NOK","NZD","SEK","SGD"
,"CAD","GBP")] <- "Others"
ksp.new1$currency <- as.factor(ksp.new1$currency)
prop.table(table(ksp.new1$currency))
```

```
##
##    Others       USD
## 0.2115444 0.7884556
```

```
barplot(prop.table(table(ksp.new1$currency)))
```

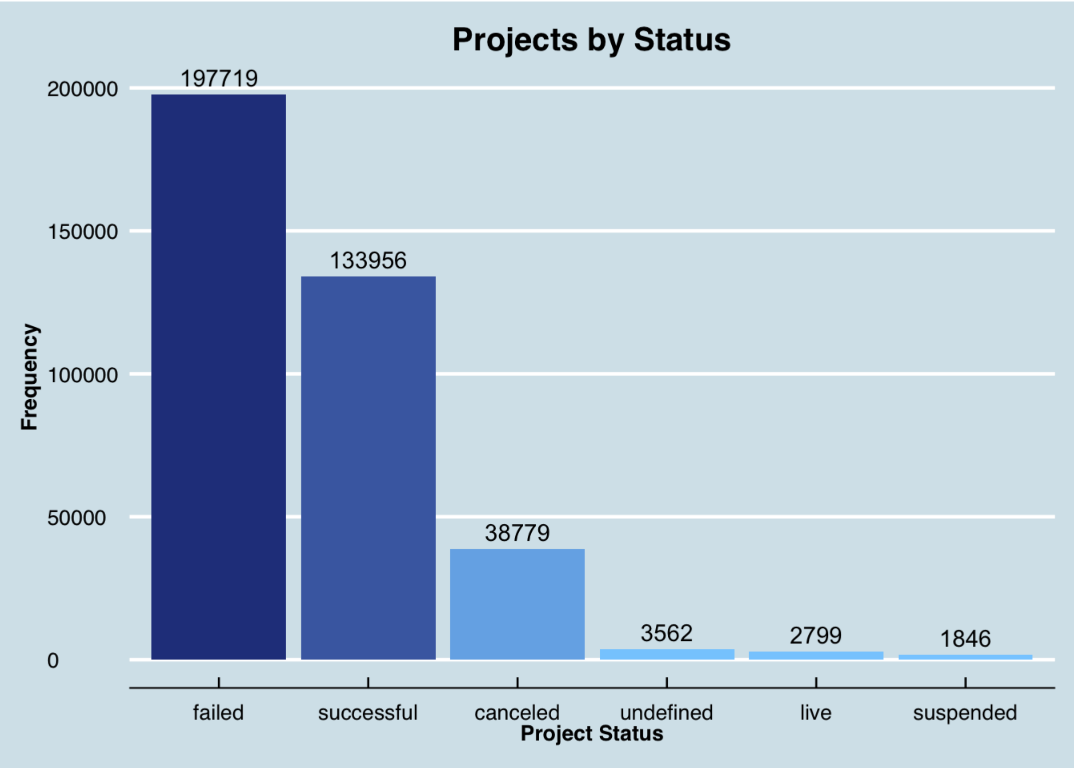


```
#approximately 80% of projects are held in US and 20% are held in other countries
```

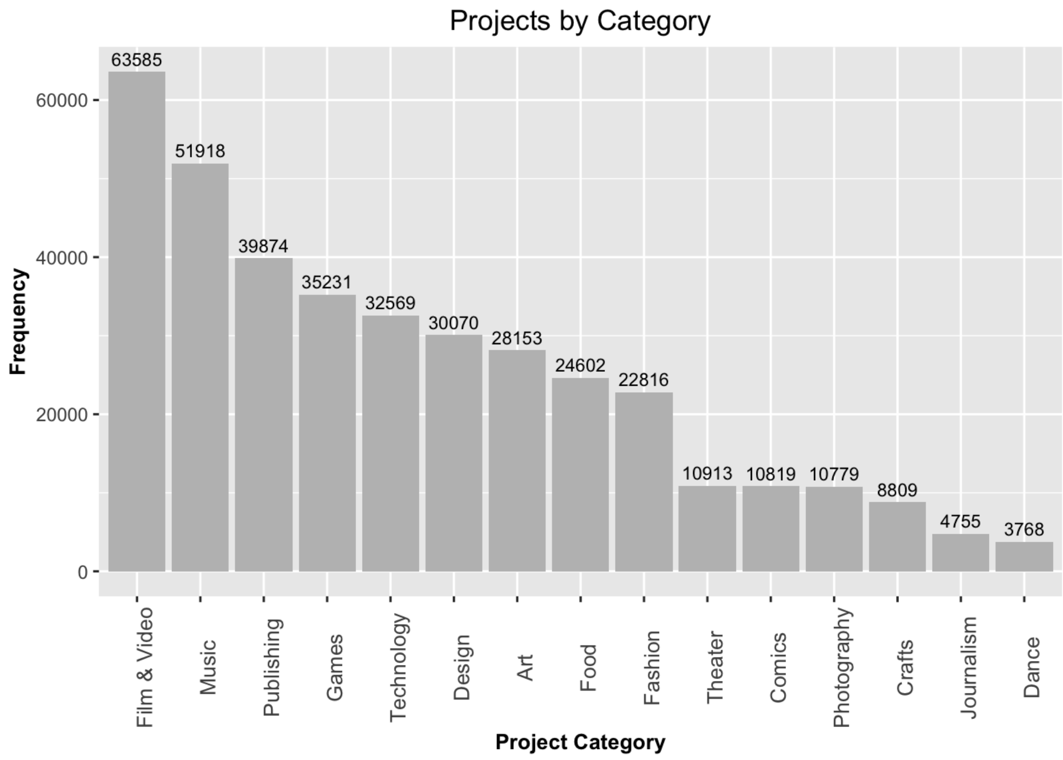
```
state.freq <- ksp %>%
  group_by(state) %>%
  summarize(count=n()) %>%
  arrange(desc(count))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

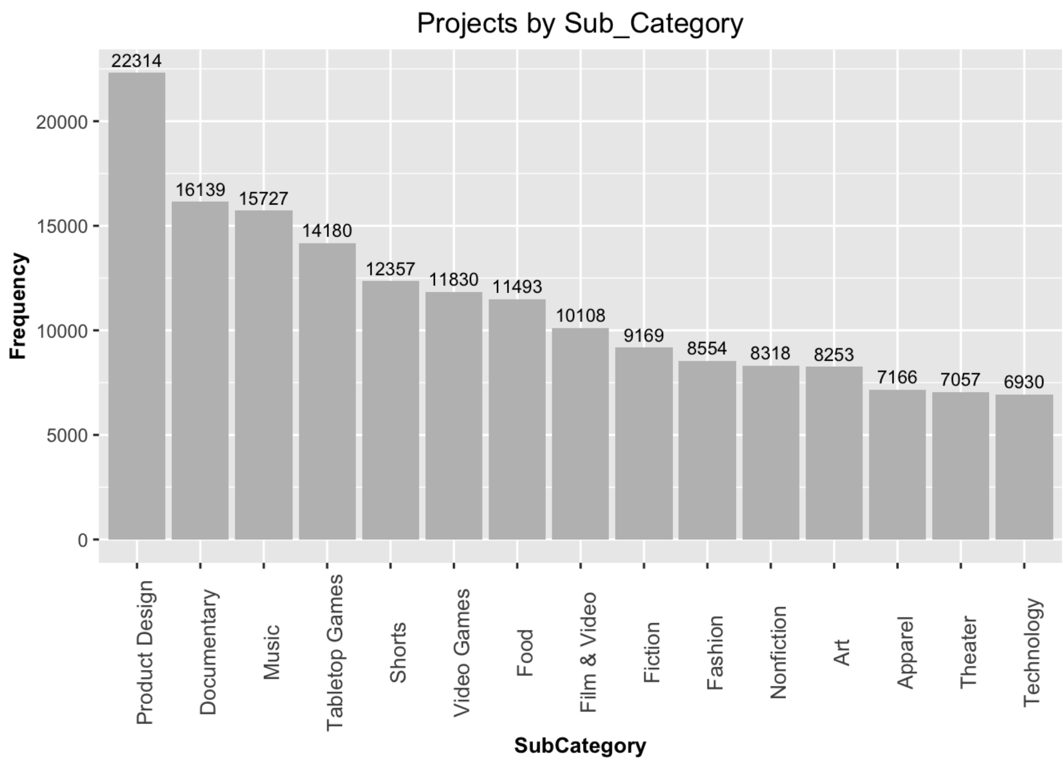
```
state.freq$state <- factor(state.freq$state, levels=state.freq$state)

ggplot(state.freq, aes(state, count, fill=count)) + geom_bar(stat="identity") +
  ggtitle("Projects by Status") + xlab("Project Status") + ylab("Frequency") +
  geom_text(aes(label=count), vjust=-0.5)  + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=10, face="bold"),
        axis.text.x=element_text(size=10), legend.position="null") +
  scale_fill_gradient(low="skyblue1", high="royalblue4")
```




```
#Below graph shows the popularity of each category

cat.freq <- ksp %>%
  group_by(main_category) %>%
  summarize(count=n()) %>%
  arrange(desc(count))

cat.freq$main_category <- factor(cat.freq$main_category, levels=cat.freq$main_category)


ggplot(cat.freq, aes(main_category, count, fill=count)) + geom_bar(stat="identity") +
    ggtitle("Projects by Category") + xlab("Project Category") + ylab("Frequency") +
    geom_text(aes(label=count), vjust=-0.5, size=3) +
    theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=10, face="bold"),
          axis.text.x=element_text(size=10, angle=90), legend.position="null") +
    scale_fill_gradient(low="grey", high="grey")
```




```r
#Below graph shows the top 15 popular sub-category projects
subcat.freq <- ksp %>%
  group_by(category) %>%
  summarize(count=n()) %>%
  arrange(desc(count))

subcat.freq$category <- factor(subcat.freq$category, levels=subcat.freq$category)

ggplot(head(subcat.freq, 15), aes(category, count, fill=count)) + geom_bar(stat="identity") +
    ggtitle("Projects by Sub_Category") + xlab("SubCategory") + ylab("Frequency") +
    geom_text(aes(label=count), vjust=-0.5, size =3) +
    theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=10, face="bold"),
          axis.text.x=element_text(size=10, angle=90), legend.position="null") +
    scale_fill_gradient(low="grey", high="grey")
```



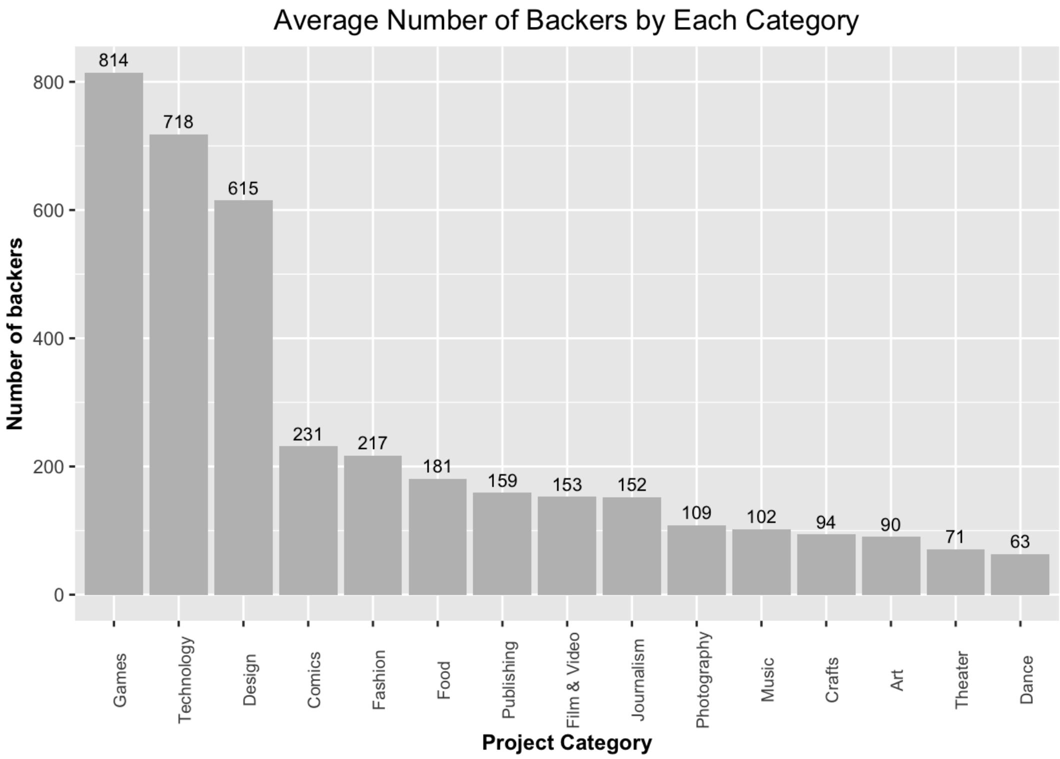
```r
#Below table shows the projects that pledged the highest amount of crowd funding.

kable(head(ksp[order(-ksp$usd_pledged_real), c(2,3,14)], 15))
```

	name	category	usd_pledged_real
157271	Pebble Time - Awesome Smartwatch, No Compromises	Product Design	20338986
250255	COOLEST COOLER: 21st Century Cooler that's Actually Cooler	Product Design	13285226
216630	Pebble 2, Time 2 + All-New Pebble Core	Product Design	12779843
289916	Kingdom Death: Monster 1.5	Tabletop Games	12393140
282417	Pebble: E-Paper Watch for iPhone and Android	Product Design	10266846
293862	The World's Best TRAVEL JACKET with 15 Features || BAUBAX	Product Design	9192056
187653	Exploding Kittens	Tabletop Games	8782572
6666	OUYA: A New Kind of Video Game Console	Gaming Hardware	8596475
309631	THE 7th CONTINENT – What Goes Up, Must Come Down.	Tabletop Games	7072757
271277	The Everyday Backpack, Tote, and Sling	Product Design	6565782
75901	Fidget Cube: A Vinyl Desk Toy	Product Design	6465690
368574	Shenmue 3	Video Games	6333296
30042	Pono Music - Where Your Soul Rediscovers Music	Sound	6225355
89482	Bring Back MYSTERY SCIENCE THEATER 3000	Television	5764229
148586	The Veronica Mars Movie Project	Narrative Film	5702153

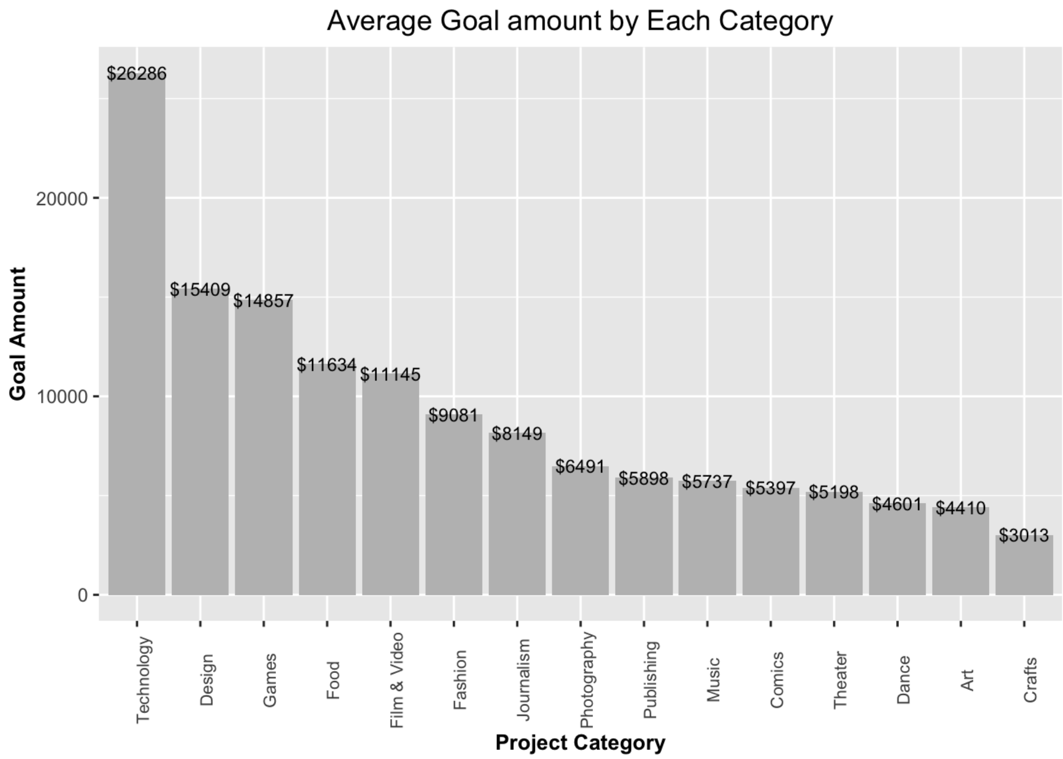
```r
#Below table shows that projects had highest number of backers.

kable(head(ksp[order(-ksp$backers), c(2,3,11)], 15))
```

	name	category	backers
187653	Exploding Kittens	Tabletop Games	219382
75901	Fidget Cube: A Vinyl Desk Toy	Product Design	154926
292245	Bring Reading Rainbow Back for Every Child, Everywhere!	Web	105857
148586	The Veronica Mars Movie Project	Narrative Film	91585
182658	Double Fine Adventure	Video Games	87142
23405	Bears vs Babies - A Card Game	Tabletop Games	85581
157271	Pebble Time - Awesome Smartwatch, No Compromises	Product Design	78471
239176	Torment: Tides of Numenera	Video Games	74405
272925	Project Eternity	Video Games	73986
38292	Yooka-Laylee - A 3D Platformer Rare-vival!	Video Games	73206
215085	ZNAPS -The $9 Magnetic Adapter for your mobile devices	Technology	70122
368574	Shenmue 3	Video Games	69320
282417	Pebble: E-Paper Watch for iPhone and Android	Product Design	68929
293644	Mighty No. 9	Video Games	67226
216630	Pebble 2, Time 2 + All-New Pebble Core	Product Design	66673

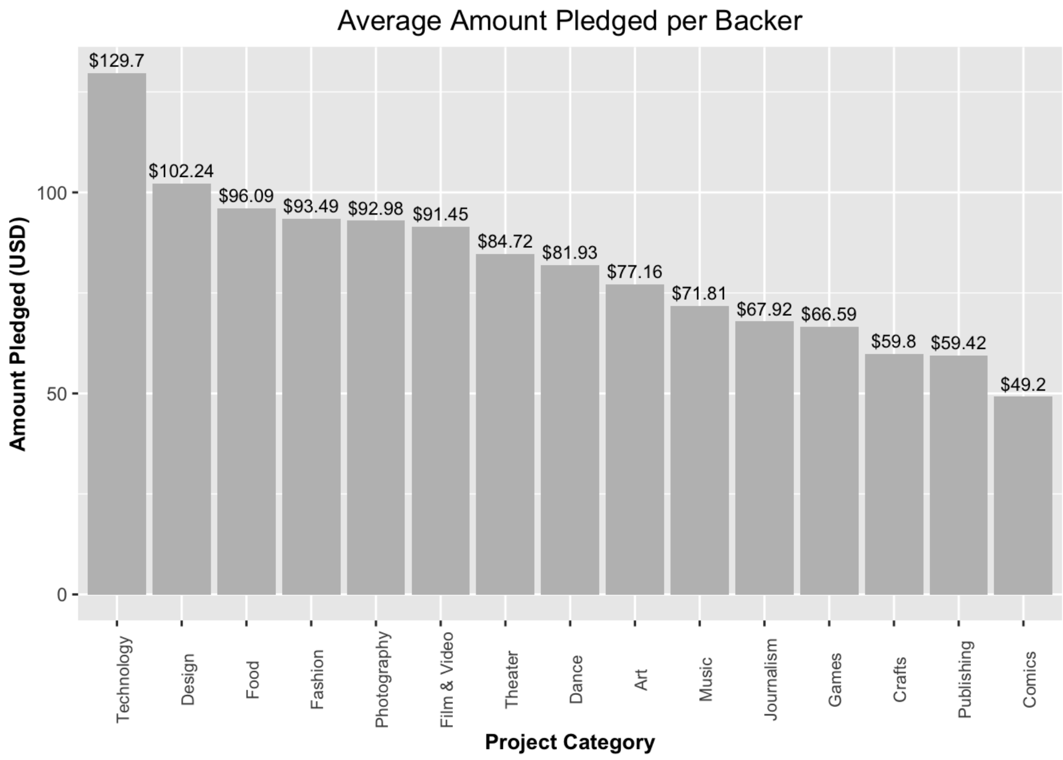
```r
# This illustrate the average number backers in each category that projects are successful.

backers.tot <- ksp %>%
  filter(state %in% c("successful")) %>%
  group_by(main_category) %>%
  summarize(project=n(), backers=sum(backers)) %>%
  mutate(total=backers/project) %>%
  arrange(desc(total))

backers.tot$main_category <- factor(backers.tot$main_category, levels=backers.tot$main_category)

ggplot(backers.tot, aes(main_category, total, fill=total)) + geom_bar(stat="identity") +
  ggtitle("Average Number of Backers by Each Category") + xlab("Project Category") +
  ylab("Number of backers") + geom_text(aes(label=round(total), vjust=-0.5), size = 3) +
    theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=10, face="bold"),
          axis.text.x=element_text(size=8, angle=90), legend.position="null") +
    scale_fill_gradient(low="grey", high="grey")
```



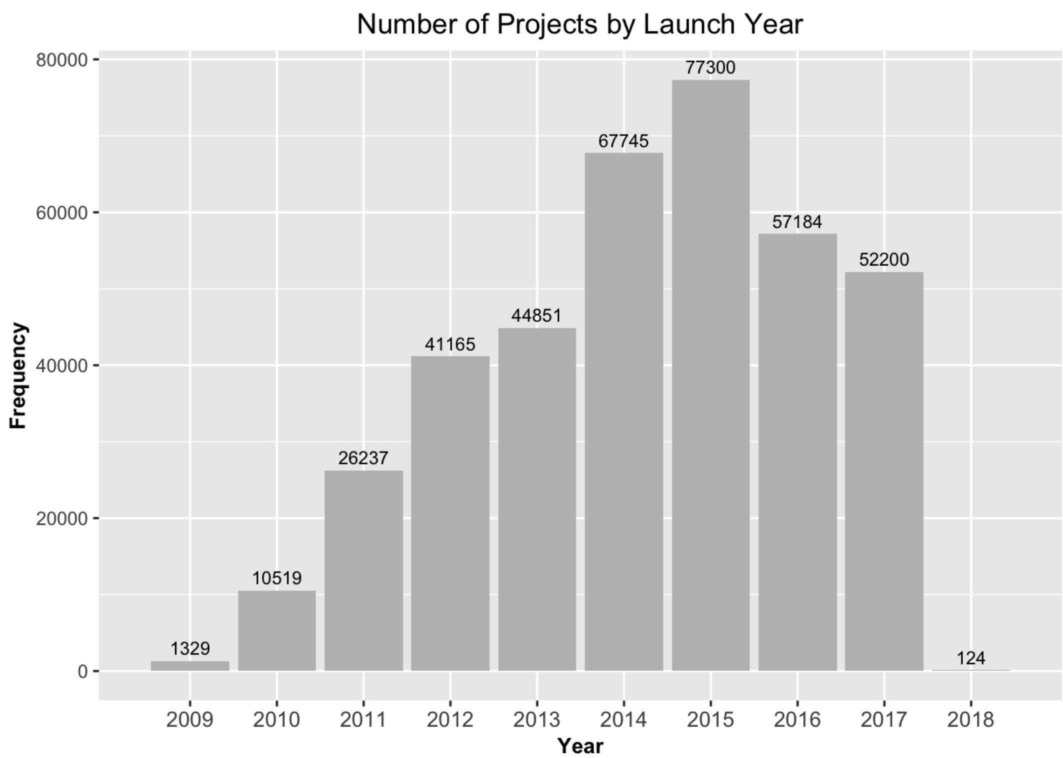

```r
#Below graph shows the average goal amount in each category where projects are successful.

Goal.tot <- ksp %>%
  filter(state %in% c("successful")) %>%
  group_by(main_category) %>%
  summarize(goal=sum(usd_goal_real), project=n()) %>%
  mutate(total = goal/project)  %>%
  arrange(desc(total))

Goal.tot$main_category <- factor(Goal.tot$main_category, levels=Goal.tot$main_category)

ggplot(Goal.tot, aes(main_category, total, fill=total)) + geom_bar(stat="identity") +
       ggtitle("Average Goal amount by Each Category") + xlab("Project Category") +ylab("Goal Amount")  +
geom_text(aes(label=paste0("$", round(total))), size=3) + theme(plot.title=element_text(hjust=0.5), axis.tit
le=element_text(size=10, face="bold"),
          axis.text.x=element_text(size=8, angle=90), legend.position="null") +
    scale_fill_gradient(low="grey", high="grey")
```



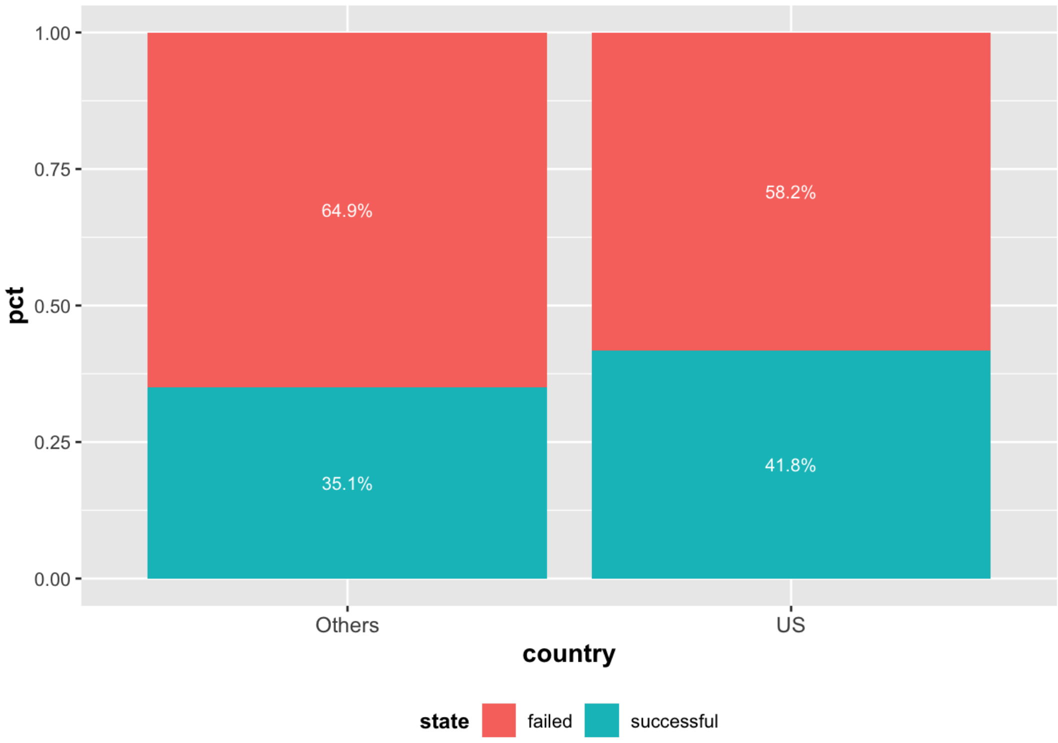

```
#Below graph shows that average pledged amount per backers where projects are successful.

pledged.avg <- ksp %>%
  filter(state %in% c("successful")) %>%
  group_by(main_category) %>%
  summarize(pledged=sum(usd_pledged_real), backers=sum(backers)) %>%
  mutate(avg=pledged/backers) %>%
  arrange(desc(avg))

pledged.avg$main_category <- factor(pledged.avg$main_category, levels=pledged.avg$main_category)

ggplot(pledged.avg, aes(main_category, avg, fill=avg)) + geom_bar(stat="identity") +
  ggtitle("Average Amount Pledged per Backer") + xlab("Project Category") +
  ylab("Amount Pledged (USD)") +
  geom_text(aes(label=paste0("$", round(avg,2))), vjust=-0.5,size=3)  + theme(plot.title=element_text(hjust=
0.5), axis.title=element_text(size=10, face="bold"),
        axis.text.x=element_text(size=8, angle=90), legend.position="null") +
    scale_fill_gradient(low="grey", high="grey")
```



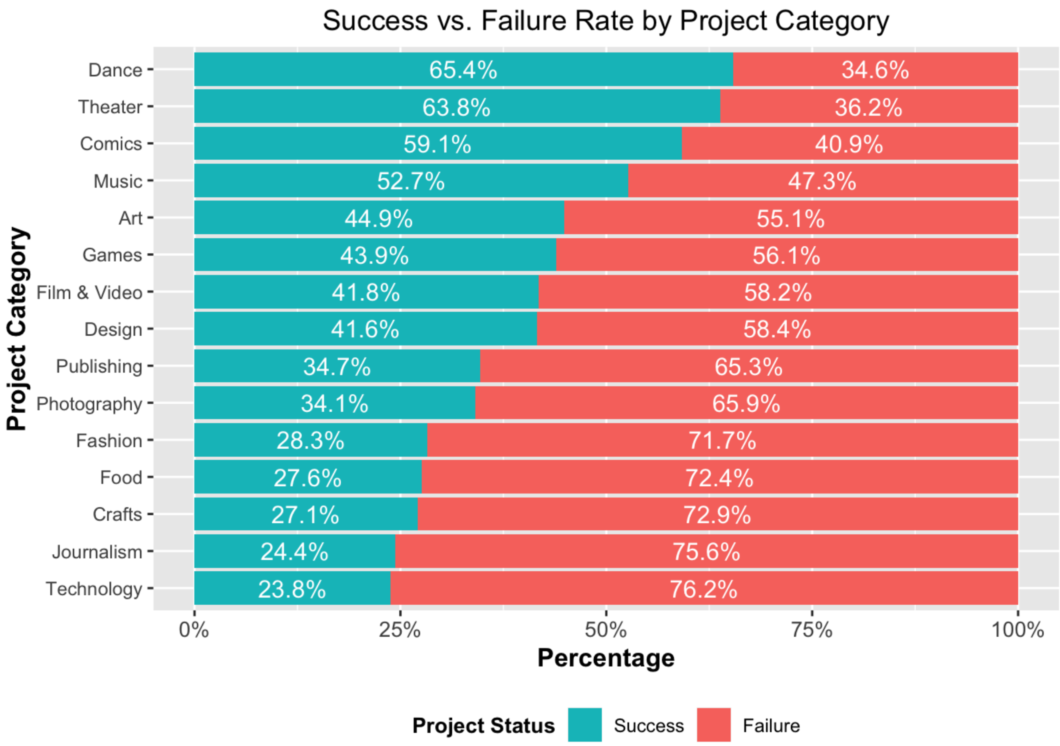

```
year.freq <- ksp %>%
  filter(year(launched)!="1970") %>%
  group_by(year=year(launched)) %>%
  summarize(count=n())

ggplot(year.freq, aes(year, count, fill=count)) + geom_bar(stat="identity") +
  ggtitle("Number of Projects by Launch Year") + xlab("Year") + ylab("Frequency") +
  scale_x_discrete(limits=c(2009:2018)) +
  geom_text(aes(label=paste0(count)), vjust=-0.5, size= 3)  +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=10, face="bold"),
        axis.text.x=element_text(size=10), legend.position="null") +
  scale_fill_gradient(low="grey", high="grey")
```



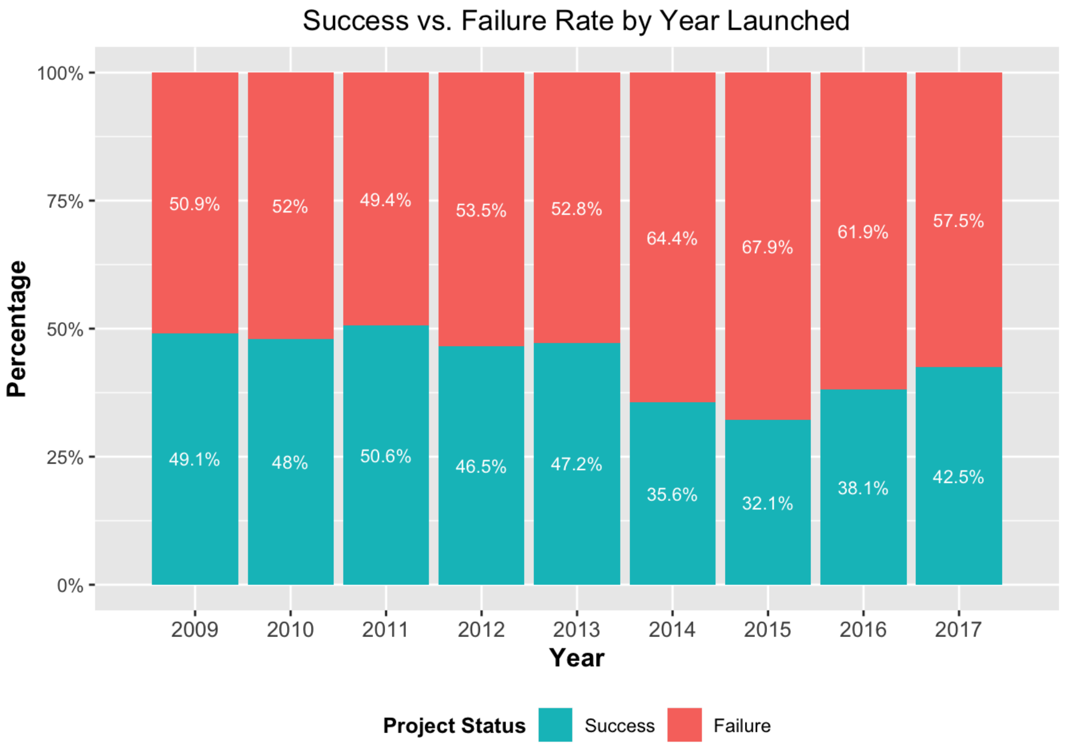
```
country.freq <- ksp.new1 %>%
  filter(state %in% c("successful", "failed")) %>%
  group_by(country, state) %>%
  summarize(count=n()) %>%
  mutate(pct=count/sum(count)) %>%
  arrange(desc(state))

ggplot(country.freq, aes(country, pct, fill =state))  + geom_bar(stat="identity")  + geom_text(aes(label=pas
te0(round(pct*100,1),"%")), position=position_stack(vjust=0.5),
            colour="white", size=3) + theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size
=12, face="bold"),
        axis.text.x=element_text(size=10), legend.position="bottom",
        legend.title=element_text(size=10, face="bold"))
```




```
state.pct <- ksp %>%
  filter(state %in% c("successful", "failed")) %>%
  group_by(main_category, state) %>%
  summarize(count=n()) %>%
  mutate(pct=count/sum(count)) %>%
  arrange(desc(state), pct)

state.pct$main_category <- factor(state.pct$main_category,
                                  levels=state.pct$main_category[1:(nrow(state.pct)/2)])

ggplot(state.pct, aes(main_category, pct, fill=state)) + geom_bar(stat="identity") +
  ggtitle("Success vs. Failure Rate by Project Category") +
  xlab("Project Category") + ylab("Percentage") + scale_y_continuous(labels=scales::percent) +
  scale_fill_discrete(name="Project Status", breaks=c("successful", "failed"),
                      labels=c("Success", "Failure")) +
  geom_text(aes(label=paste0(round(pct*100,1),"%")), position=position_stack(vjust=0.5),
            colour="white", size=4)  +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=10), legend.position="bottom",
        legend.title=element_text(size=10, face="bold")) + coord_flip()
```

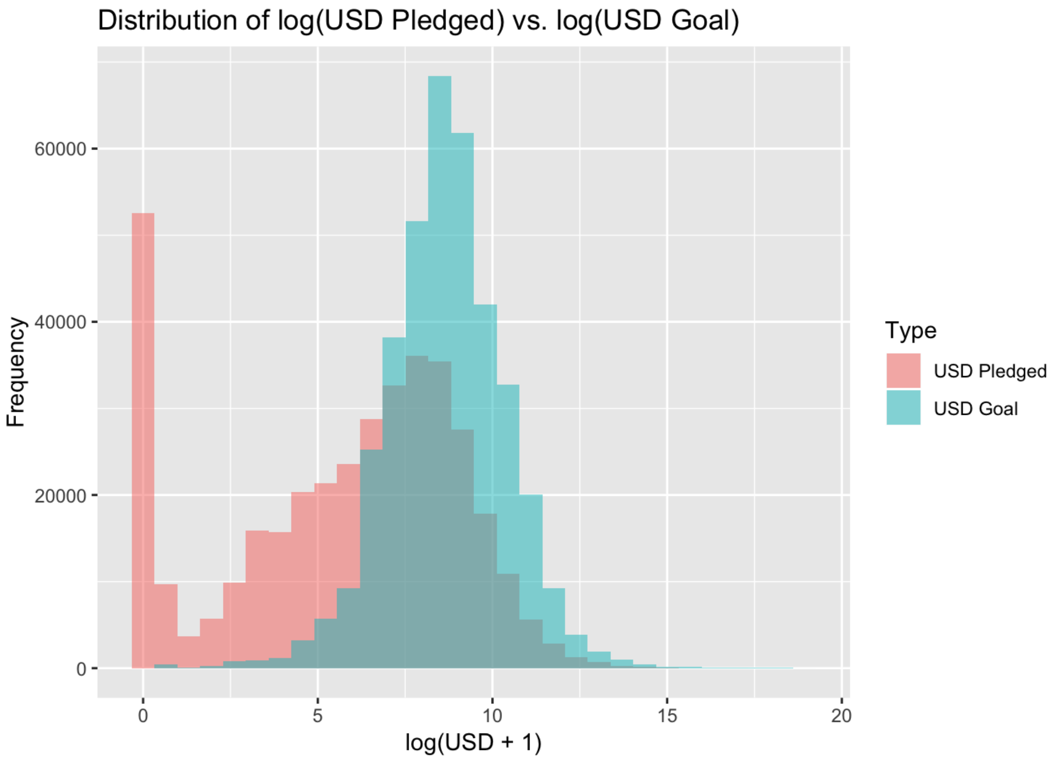



```r
state.pct2 <- ksp %>%
  filter(year(launched)!="1970", state %in% c("successful", "failed")) %>%
  group_by(year=year(launched), state) %>%
  summarize(count=n()) %>%
  mutate(pct=count/sum(count)) %>%
  arrange(desc(state))

ggplot(state.pct2, aes(year, pct, fill=state)) + geom_bar(stat="identity") +
  ggtitle("Success vs. Failure Rate by Year Launched") +
  xlab("Year") + ylab("Percentage") + scale_x_discrete(limits=c(2009:2017)) +
  scale_y_continuous(labels=scales::percent) +
  scale_fill_discrete(name="Project Status", breaks=c("successful", "failed"),
                      labels=c("Success", "Failure")) +
  geom_text(aes(label=paste0(round(pct*100,1),"%")), position=position_stack(vjust=0.5),
            colour="white", size=3)  +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=10), legend.position="bottom",
        legend.title=element_text(size=10, face="bold"))
```

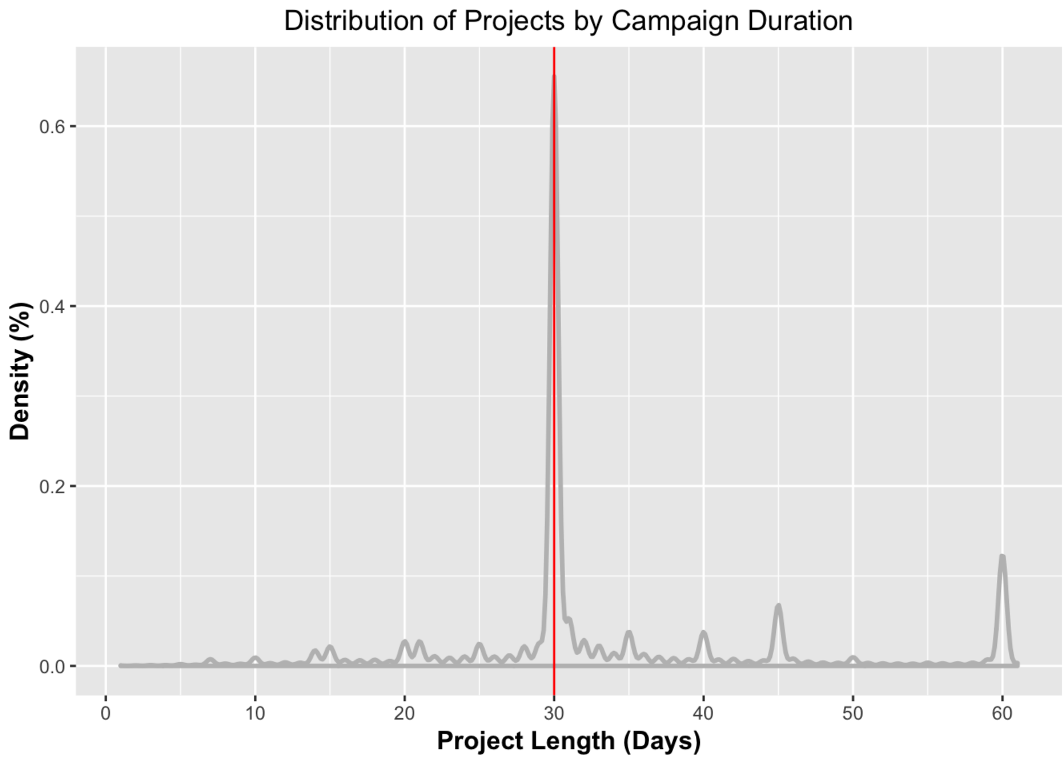



```
usd.amounts <- gather(ksp, type, amount, usd_pledged_real, usd_goal_real, factor_key=T)

ggplot(usd.amounts, aes(log(amount+1), fill=type)) +
  geom_histogram(alpha=0.5, position="identity") +
  ggtitle("Distribution of log(USD Pledged) vs. log(USD Goal)") + xlab("log(USD + 1)") +
  ylab("Frequency") + scale_fill_discrete("Type", labels=c("USD Pledged", "USD Goal"))
```

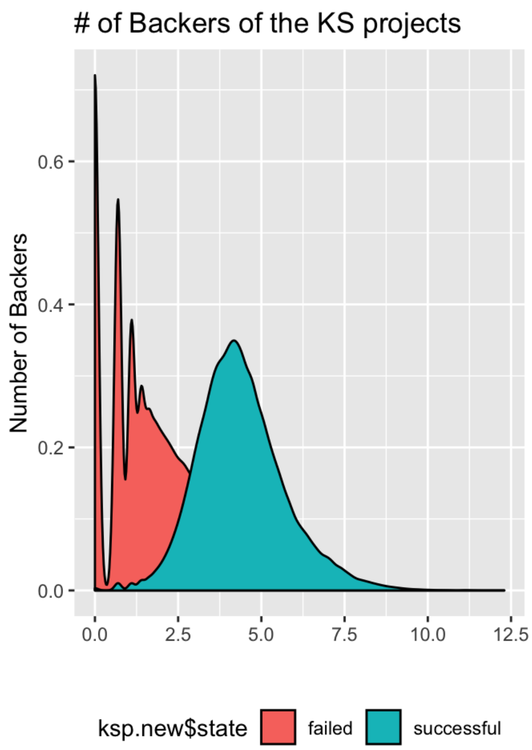
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

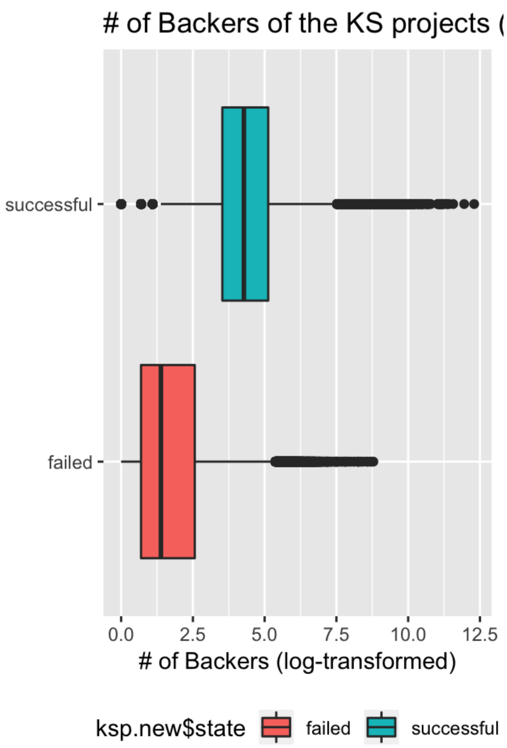



```
ggplot(ksp.new[ksp.new$duration <= 61,], aes(duration)) + geom_density(colour="grey", size=1) +
  ggtitle("Distribution of Projects by Campaign Duration") + xlab("Project Length (Days)") +
  ylab("Density (%)") + scale_x_continuous(breaks=c(0,10,20,30,40,50,60)) +
  geom_vline(xintercept=30, colour="red") +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"))
```

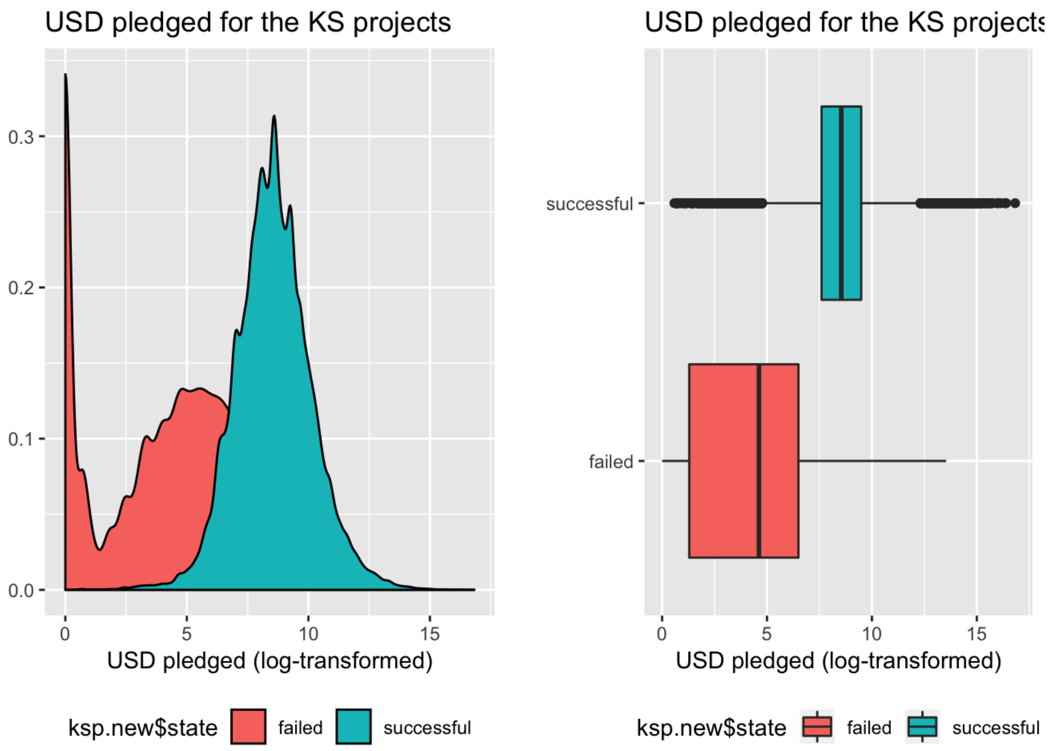



```
p1 <- ggplot(ksp.new, aes(log(backers+1),  fill = ksp.new$state)) +
  geom_density() +
  theme(legend.position = "bottom") +
  ylab("Number of Backers") + xlab("") +
  ggtitle("# of Backers of the KS projects")

p2 <- ggplot(ksp.new, aes(x = state, y = log(backers+1), fill = ksp.new$state)) +
  geom_boxplot() +
  coord_flip() +
  theme(legend.position = "bottom") +
  ylab("# of Backers (log-transformed)") + xlab("") +
  ggtitle("# of Backers of the KS projects (Log)")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```

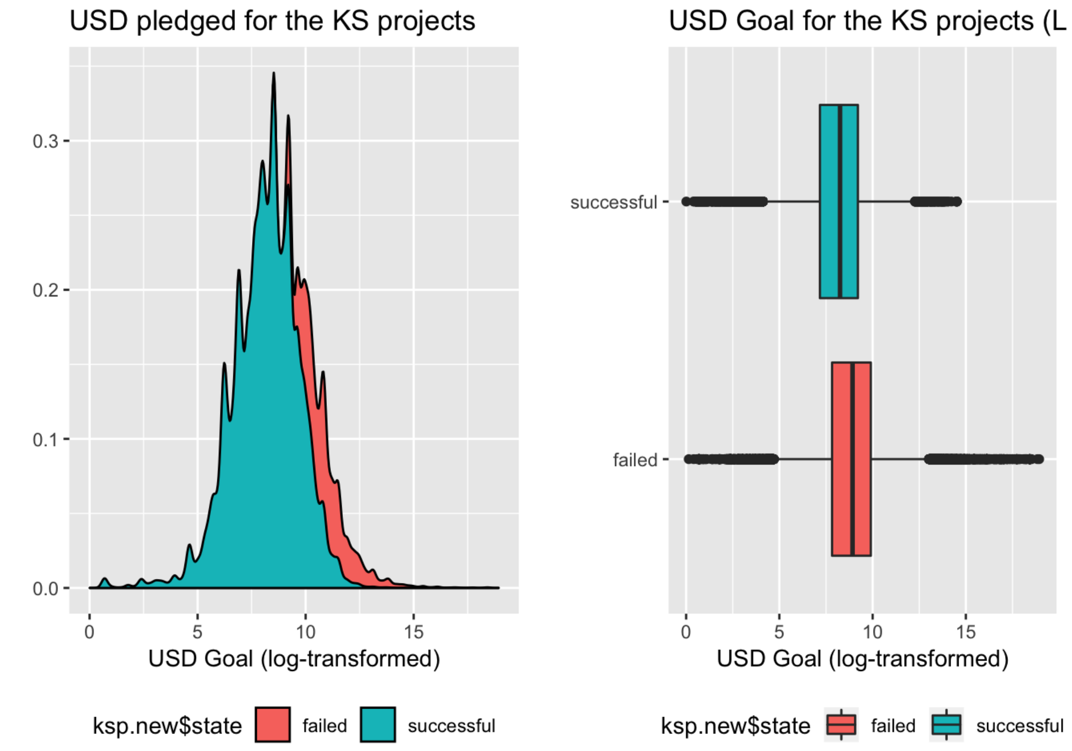




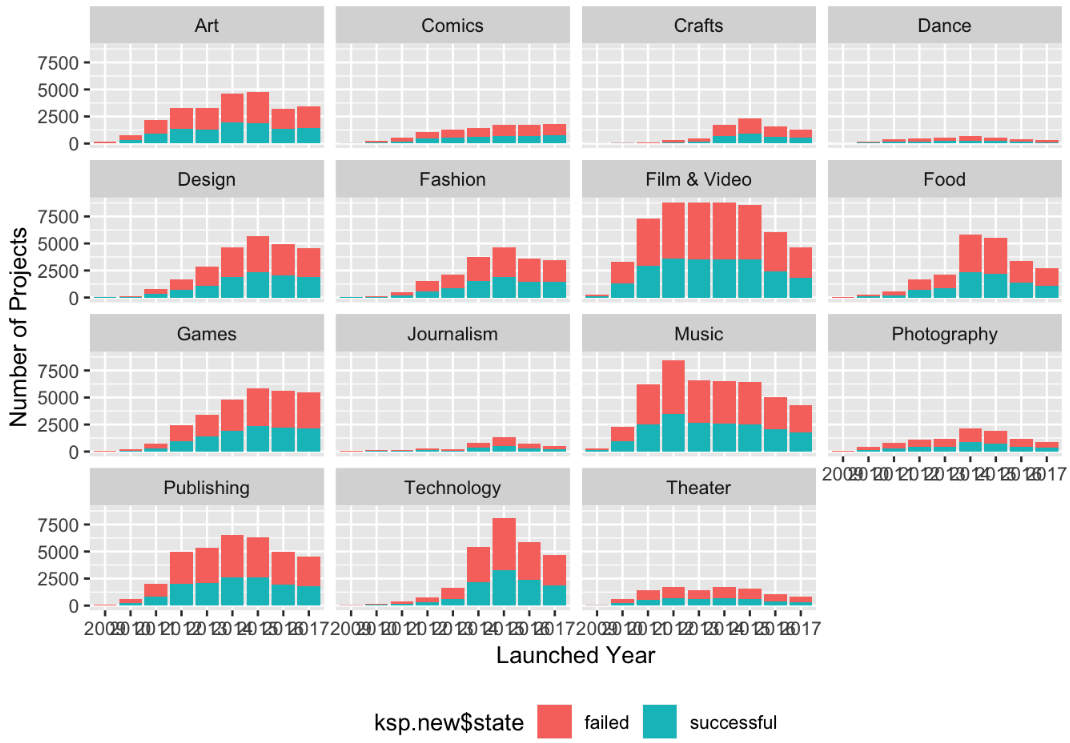
```
p1 <- ggplot(ksp.new, aes(log(usd_pledged_real+1),  fill = ksp.new$state)) +
  geom_density() +
  theme(legend.position = "bottom") +
  xlab("USD pledged (log-transformed)") + ylab("") +
  ggtitle("USD pledged for the KS projects")

# Log-transformed usd_pledged_real
p2 <- ggplot(ksp.new, aes(x = state, y = log(usd_pledged_real+1), fill = ksp.new$state)) +
  geom_boxplot() +
  theme(legend.position = "bottom") +
  ylab("USD pledged (log-transformed)") + xlab("") +
  scale_y_continuous(labels = scales::comma) +
  coord_flip() +
  ggtitle("USD pledged for the KS projects (Log)")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```




```
p1 <- ggplot(ksp.new, aes(log(usd_goal_real+1),  fill = ksp.new$state)) +
  geom_density() +
  theme(legend.position = "bottom") +
  xlab("USD Goal (log-transformed)") + ylab("") +
  ggtitle("USD pledged for the KS projects")

# Log-transformed usd_pledged_real
p2 <- ggplot(ksp.new, aes(x = state, y = log(usd_goal_real+1), fill = ksp.new$state)) +
  geom_boxplot() +
  theme(legend.position = "bottom") +
  ylab("USD Goal (log-transformed)") + xlab("") +
  scale_y_continuous(labels = scales::comma) +
  coord_flip() +
  ggtitle("USD Goal for the KS projects (Log)")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```



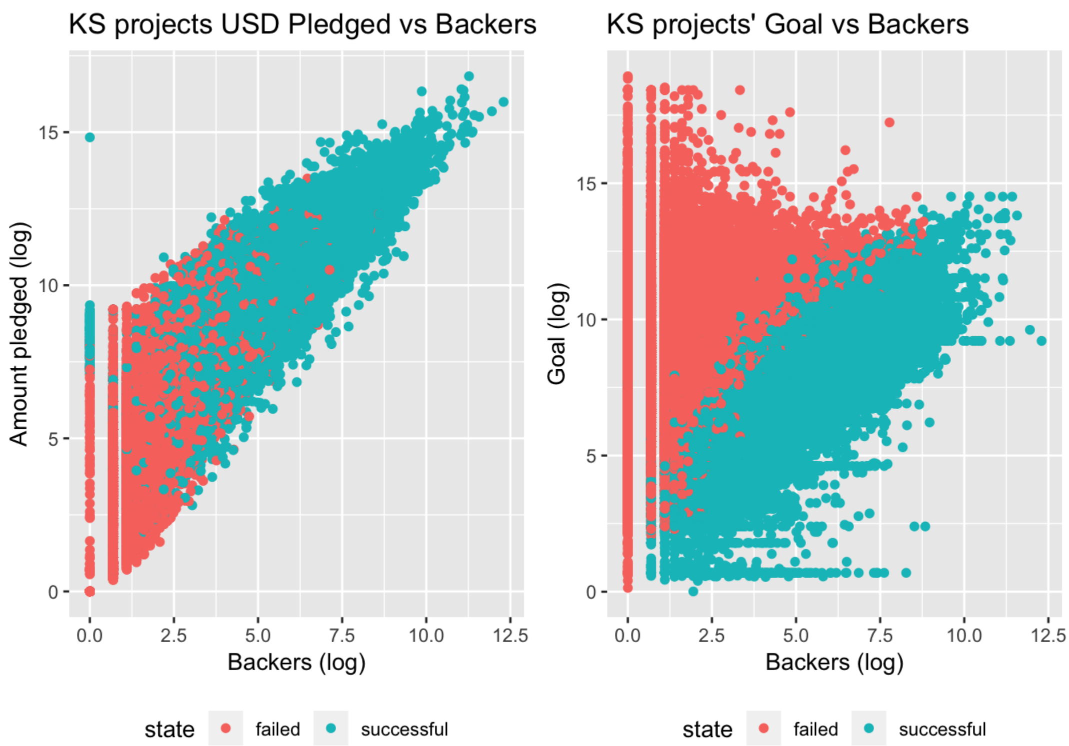

```
ggplot(ksp.new, aes(launched_year, fill = ksp.new$state)) +
  geom_bar() +
  theme(legend.position = "bottom") +
  facet_wrap( ~ main_category) +
  ylab("Number of Projects") + xlab("Launched Year")
```




```
  ggtitle("KS projects launched over time by Category")
```

```
## $title
## [1] "KS projects launched over time by Category"
##
## $subtitle
## NULL
##
## attr(,"class")
## [1] "labels"
```

```
p1 <- ggplot(ksp.new, aes(x = log(backers+1), y = log(usd_pledged_real+1))) +
  geom_jitter(aes(color = state)) +
  theme(legend.position = "bottom") +
  ylab("Amount pledged (log)") + xlab("Backers (log)") +
  ggtitle("KS projects USD Pledged vs Backers")

# 4. Goal vs Backers
p2 <- ggplot(ksp.new, aes(x = log(backers+1), y = log(usd_goal_real+1))) +
  geom_jitter(aes(color = state)) +
  theme(legend.position = "bottom") +
  ylab("Goal (log)") + xlab("Backers (log)") +
  ggtitle("KS projects' Goal vs Backers")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```

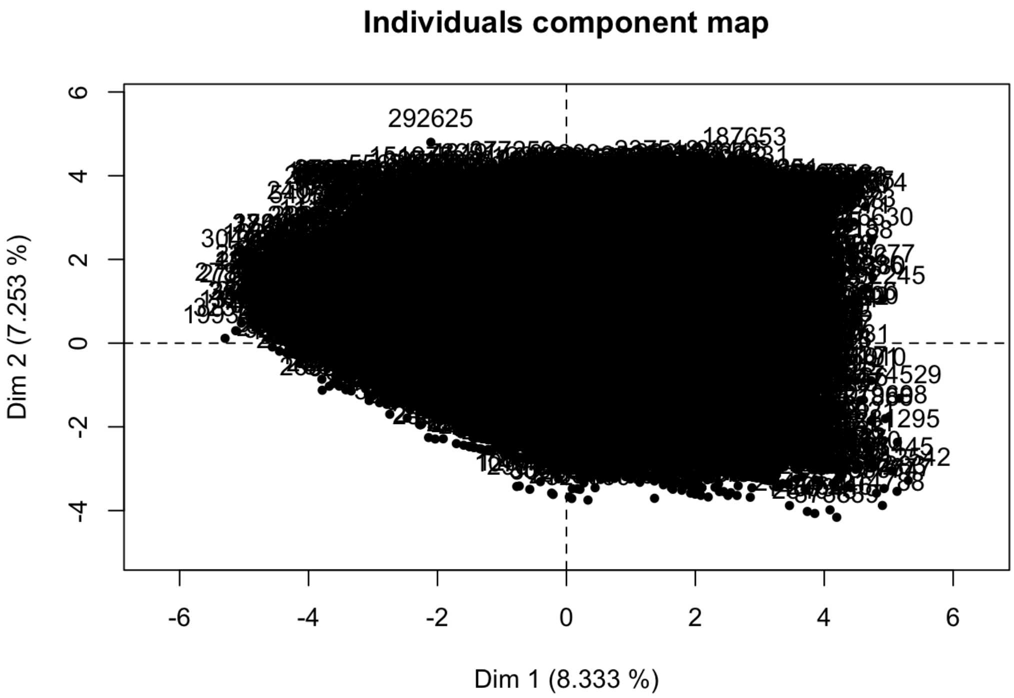


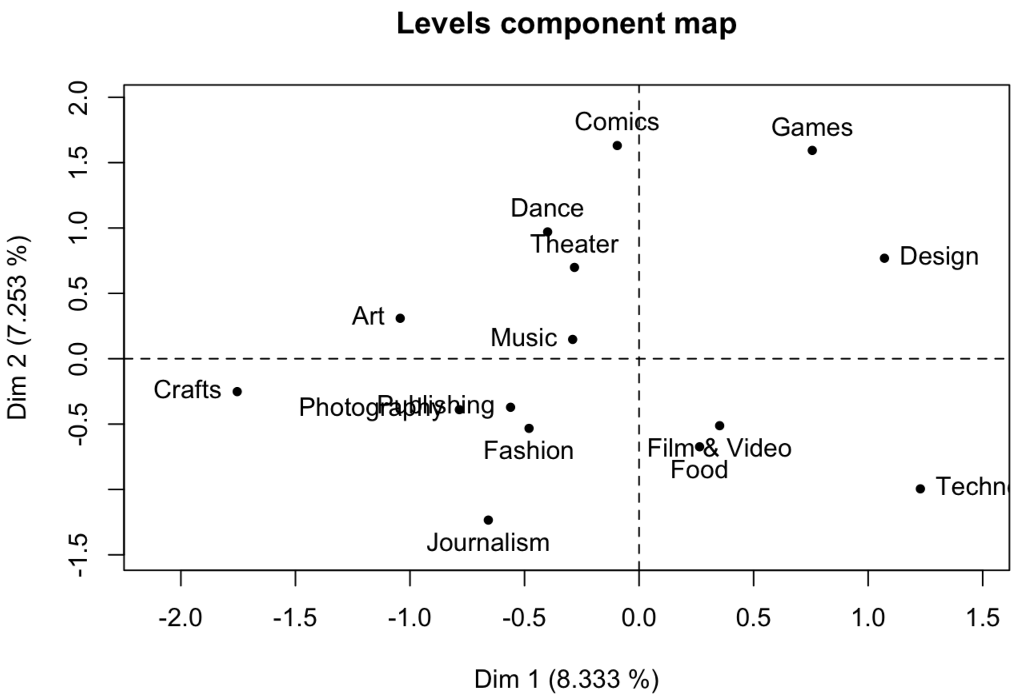
Data split into training/test

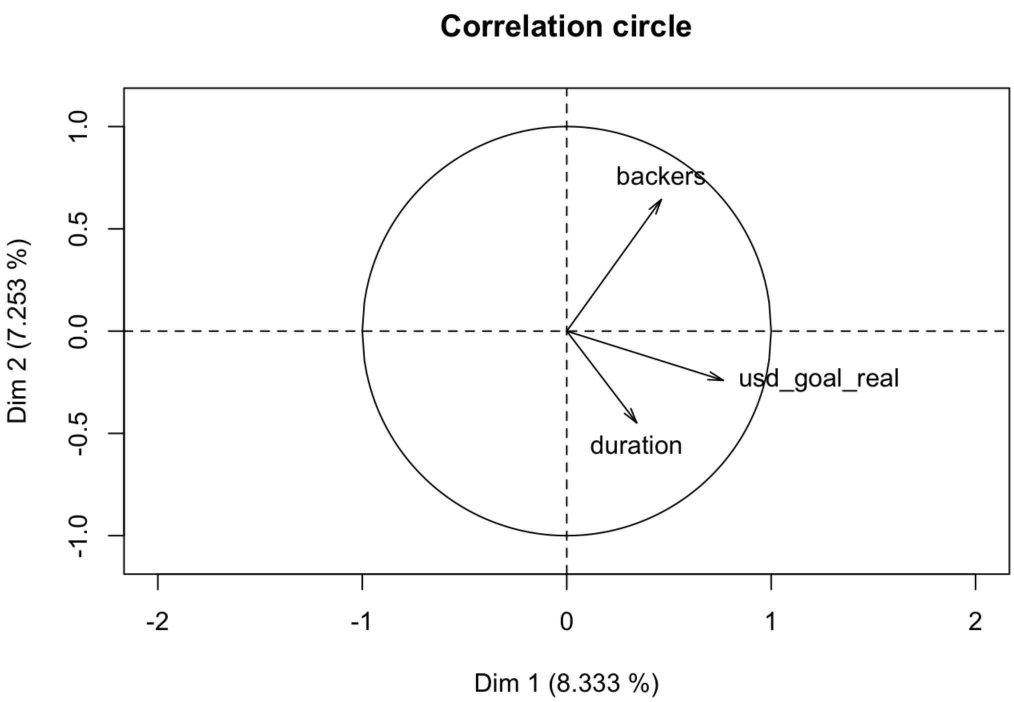
```
kspN <- ksp.new1[, c(4,9,10,11,13,14,15)]

kspN <- kspN[kspN$currency == 'USD' & kspN$country == 'US',]

kspN <- kspN[,-2:-3]

kspN$backers <- log(kspN$backers+1)
kspN$usd_goal_real <- log(kspN$usd_goal_real+1)

normalize <- function(x) {
               return ((x - min(x)) / (max(x) - min(x))) }

kspN[,2:4] <- lapply(kspN[,2:4], normalize)

rn_train <- sample(nrow(kspN), floor(nrow(kspN)*0.7))
ksp.train <- kspN[rn_train,]
ksp.test <- kspN[-rn_train,]
#subsetting dataset which has a contribution to the target variable
#Splitting dataset into training and test with 7:3 ratio
#logarithm and normalization is used for data normalization
```

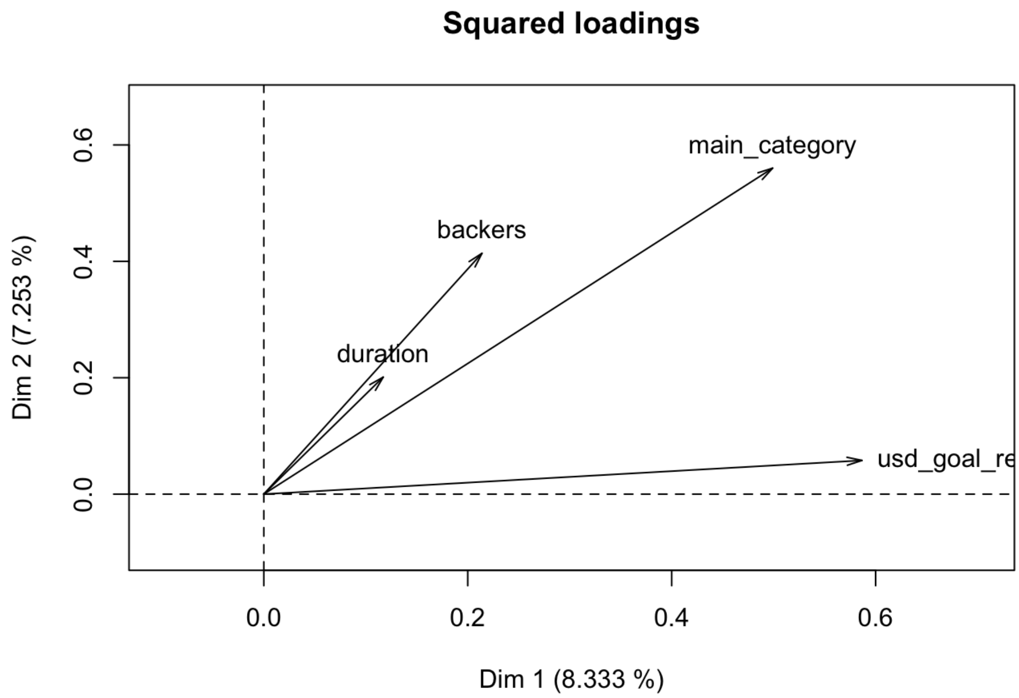
PCA

```
kspN.split <- splitmix(kspN[,-5])

X1 <- kspN.split$X.quanti
X2 <- kspN.split$X.quali

res.pcamix <- PCAmix(X.quanti=X1, X.quali=X2, rename.level=TRUE ,ndim = 5 , graph=FALSE)

obj <- PCAmix(X.quanti = X1, X.quali = X2, ndim =2)
```




Levels component map
Comics
Games
Dance
Theater
Design
Art
Music
Crafts
Photography
Publishing
Fashion
Film & Video
Food
Techno
Journalism
Dim 2 (7.253 %)
Dim 1 (8.333 %)
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0


Correlation circle
backers
usd_goal_real
duration
Dim 2 (7.253 %)
Dim 1 (8.333 %)
-2
-1
0
1
2
-1.0
-0.5
0.0
0.5
1.0




```
res.pcamix$sqload
```

```
##                    dim 1      dim 2       dim 3        dim 4        dim 5
## backers        0.2139768 0.41386223 0.005502857 3.455019e-27 1.155156e-30
## usd_goal_real  0.5866118 0.05794274 0.017115492 1.071778e-27 3.000147e-28
## duration       0.1171113 0.20108375 0.219660796 1.563008e-26 4.127902e-28
## main_category  0.4989763 0.56014433 0.807751003 1.000000e+00 1.000000e+00
```

```
ksp.new2 <- data.frame(model.matrix(~.-1, data=kspN))
ksp.new2 <- ksp.new2[,-19]
ksp.pca.normdata <- prcomp(ksp.new2, scale=TRUE, center=TRUE)
ksp.pca.normdata$rotation
```

```
##                                   PC1         PC2          PC3
## main_categoryArt          -0.27094172  0.07047385 -0.201598041
## main_categoryComics       -0.01988496  0.21838267 -0.050391374
## main_categoryCrafts       -0.20222510 -0.04045887 -0.093717847
## main_categoryDance        -0.03445209  0.07212634 -0.041402298
## main_categoryDesign        0.24588505  0.23128494 -0.053792285
## main_categoryFashion      -0.09583030 -0.09735865 -0.106905194
## main_categoryFilm...Video  0.21391216 -0.41706240 -0.144336785
## main_categoryFood          0.06060582 -0.12688355 -0.058848248
## main_categoryGames         0.18234149  0.44652655 -0.212635990
## main_categoryJournalism   -0.05291966 -0.08662001 -0.007672786
## main_categoryMusic        -0.16077063  0.16906158  0.853314188
## main_categoryPhotography  -0.10236645 -0.04468757 -0.036233960
## main_categoryPublishing   -0.19606920 -0.12855648 -0.177654406
## main_categoryTechnology    0.28473938 -0.17242711  0.038660288
## main_categoryTheater      -0.04318927  0.09703605 -0.049575474
## backers                    0.35111776  0.53023153 -0.016270542
## usd_goal_real              0.61705410 -0.15492921  0.016647571
## duration                   0.25930572 -0.29866573  0.311314883
##                                    PC4         PC5          PC6
## main_categoryArt          -0.042278397 -0.38052295 -0.383774133
## main_categoryComics       -0.093995416  0.04179531 -0.079657867
## main_categoryCrafts        0.007600837 -0.11115357  0.059676897
## main_categoryDance        -0.020031607 -0.01264512 -0.005821886
## main_categoryDesign        0.106292092  0.05201569 -0.662669587
## main_categoryFashion       0.150146180 -0.23051480  0.156764296
## main_categoryFilm...Video -0.734656260  0.06393734  0.032716586
## main_categoryFood          0.284071730 -0.27815885  0.262321103
## main_categoryGames        -0.041010381  0.04607494  0.539524890
## main_categoryJournalism    0.061597695 -0.05339884  0.010367672
```

```
## main_categoryMusic        -0.073284036  0.01514449  0.077704907
## main_categoryPhotography   0.039189583 -0.07164247  0.006471647
## main_categoryPublishing    0.306585599  0.80078283 -0.012077662
## main_categoryTechnology    0.430041946 -0.19361762 -0.005658479
## main_categoryTheater      -0.020271174 -0.01767342 -0.032363059
## backers                   -0.133720022  0.06983598 -0.030183002
## usd_goal_real              0.156759413 -0.01756230  0.016860276
## duration                   0.021934541  0.09519606 -0.097431936
##                                    PC7          PC8          PC9
## main_categoryArt          -0.353248055 -0.382665793 -0.313755307
## main_categoryComics        0.018126822 -0.008842175  0.627935820
## main_categoryCrafts        0.049429376  0.144360488  0.198657158
## main_categoryDance         0.014118428  0.024991480  0.126074825
## main_categoryDesign        0.352365468  0.194758158 -0.169175291
## main_categoryFashion       0.128674375  0.724794204 -0.221936107
## main_categoryFilm...Video  0.001108080 -0.002704597 -0.021051267
## main_categoryFood          0.622713677 -0.489271830  0.027986578
## main_categoryGames        -0.172042427 -0.037972041 -0.261795061
## main_categoryJournalism    0.010014051  0.040020820  0.016751486
## main_categoryMusic        -0.008053133 -0.004340543 -0.112751356
## main_categoryPhotography   0.026896720  0.082103650  0.201717883
## main_categoryPublishing   -0.048078886 -0.102802642 -0.097612652
## main_categoryTechnology   -0.556126735  0.031246109  0.232779715
## main_categoryTheater       0.023999697  0.041152011  0.425960565
## backers                   -0.004423562 -0.027923232  0.041274629
## usd_goal_real             -0.006344157 -0.005605841 -0.060413194
## duration                  -0.032265722 -0.074295705  0.007858066
##                                   PC10         PC11         PC12
## main_categoryArt          -0.185577859 -0.037178893 -0.049067987
## main_categoryComics       -0.461212802  0.427447404 -0.019928363
## main_categoryCrafts        0.547604850  0.220001606  0.525281004
## main_categoryDance         0.017548394  0.018794580  0.056368251
## main_categoryDesign        0.217592902  0.039986574  0.041840877
## main_categoryFashion      -0.415751523 -0.033864770 -0.037935707
## main_categoryFilm...Video  0.005410264  0.014123912  0.051786570
## main_categoryFood         -0.107818987  0.013200462  0.043085035
## main_categoryGames         0.154321407  0.004791753 -0.071806478
## main_categoryJournalism    0.137567890  0.005325393 -0.081536722
## main_categoryMusic         0.005692978  0.007129381  0.049300575
## main_categoryPhotography   0.393854895  0.133860986 -0.813797345
## main_categoryPublishing   -0.094894745 -0.006702799  0.020010679
## main_categoryTechnology    0.060440736  0.060203636  0.131734248
## main_categoryTheater       0.015300526 -0.860996685  0.019106912
## backers                   -0.066494381 -0.004589007  0.008142845
## usd_goal_real             -0.013642528 -0.008387560 -0.004711702
## duration                  -0.073623740 -0.030594082 -0.125142888
##                                   PC13          PC14          PC15
## main_categoryArt           0.005090264  0.0446754968 -0.2036953342
## main_categoryComics        0.157185674  0.1127349903 -0.1284595379
## main_categoryCrafts       -0.025902993  0.1801975610 -0.3222939071
## main_categoryDance        -0.345941433 -0.8970115056 -0.1702038191
## main_categoryDesign        0.027408103  0.0273259744  0.0408338429
## main_categoryFashion      -0.046782754  0.0370508726 -0.0919724349
## main_categoryFilm...Video -0.046208718  0.0074291992  0.1936007447
## main_categoryFood         -0.063514834  0.0179312557  0.0820493130
## main_categoryGames         0.111835361  0.0441334523 -0.2528578831
## main_categoryJournalism    0.880949601 -0.3570816169  0.1179007619
## main_categoryMusic        -0.035226052  0.0173305292  0.1260062359
## main_categoryPhotography  -0.179454970  0.0827003346 -0.0002166188
## main_categoryPublishing   -0.042809711  0.0219936642  0.0170826654
## main_categoryTechnology   -0.100888604 -0.0007767782  0.3056308167
## main_categoryTheater       0.032975994  0.0909024009 -0.0829647539
## backers                   -0.040712915 -0.0058264537  0.0873624743
## usd_goal_real              0.005685797  0.0074982939 -0.0771496500
## duration                   0.094859509  0.0162924031 -0.7395534019
##                                  PC16        PC17         PC18
## main_categoryArt           0.19804263  0.12746966  2.832715e-01
## main_categoryComics       -0.14073389  0.18584178  1.834381e-01
## main_categoryCrafts        0.28605968  0.04603661  1.574994e-01
## main_categoryDance        -0.01624648  0.06514851  1.138489e-01
## main_categoryDesign       -0.33341645 -0.03393806  2.704205e-01
## main_categoryFashion       0.16308831 -0.07759192  2.417434e-01
## main_categoryFilm...Video -0.01792741 -0.08551502  4.048717e-01
```

```
## main_categoryFood           0.02594230 -0.15827913  2.673423e-01
## main_categoryGames         -0.37263981  0.04454181  2.818722e-01
## main_categoryJournalism     0.15266952 -0.03787253  1.143589e-01
## main_categoryMusic          0.04885582  0.16057564  3.806407e-01
## main_categoryPhotography    0.16244279  0.02617454  1.729842e-01
## main_categoryPublishing     0.18636568  0.01378076  3.296763e-01
## main_categoryTechnology    -0.14741577 -0.28157844  2.675138e-01
## main_categoryTheater       -0.01932311  0.09201930  1.838185e-01
## backers                     0.62150309 -0.41113925 -2.400857e-15
## usd_goal_real               0.29024272  0.68957687  1.942890e-15
## duration                   -0.04281874 -0.37540599 -2.373102e-15
```

```r
head(ksp.pca.normdata$x)
```

```
##          PC1          PC2         PC3         PC4         PC5         PC6
## 2  1.7045235 -1.843006714  0.23632358 -1.66100112  0.31663897 -0.09367092
## 3  1.3011211 -1.926658304 -0.10665622 -1.54902169  0.15115540  0.04508368
## 4 -0.9623938 -0.012824931  2.25849016 -0.02579493 -0.08521202  0.27589781
## 6  1.5733521  0.011338709 -0.23586158  1.19350051 -1.03810412  1.00640143
## 7 -0.6885352  0.002152909 -0.61265611  0.97470902 -1.20131725  1.11960018
## 8  1.1932596 -0.630795277  0.01111874  1.26429136 -1.02043963  0.95153773
##            PC7         PC8         PC9         PC10         PC11
## 2 -0.063904837 -0.15154294 -0.08481251 -0.139711439 -0.033938643
## 3 -0.025018440 -0.04672892 -0.13924440 -0.009335462  0.002632357
## 4 -0.002820278  0.05180556 -0.33955964  0.120100018  0.033883305
## 6  2.444004553 -1.97300459  0.10422194 -0.535119010  0.031100386
## 7  2.502514657 -1.83562151  0.18377137 -0.326408179  0.092817104
## 8  2.425885776 -2.00255302  0.09873973 -0.525926966  0.015286157
##          PC12         PC13       PC14       PC15       PC16       PC17
## 2 -0.12594935  0.082880555 0.05329716 -1.0673273  0.1533069 -0.2258925
## 3  0.01123810  0.004794612 0.04065293 -0.2977024 -0.1816494  0.6808892
## 4  0.16064746 -0.074564095 0.04255375  0.4721205 -0.5701114  1.0022287
## 6  0.15962743 -0.284944086 0.06756007  0.2719070  1.3464613 -0.2655969
## 7  0.30414177 -0.352284496 0.03873269  1.1906545 -0.1520671 -0.9261632
## 8  0.05805703 -0.177686943 0.08215339 -0.3427591  0.6292021 -0.4768423
##            PC18
## 2  8.668940e-13
## 3  8.718789e-13
## 4 -8.032273e-13
## 6 -3.273318e-12
## 7 -3.271961e-12
## 8 -3.273804e-12
```

Feature selection

```r
null <- glm(state~1, data = ksp.train, family = "binomial")
full <- glm(state~., data = ksp.train, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
stepF <- stepAIC(null, scope=list(lower=null, upper=full), direction= "forward", trace=TRUE)
```

```
## Start:  AIC=248624.6
## state ~ 1
## 
##                 Df Deviance    AIC
## + backers        1   129430 129434
## + usd_goal_real  1   239647 239651
## + main_category 14   240112 240142
## + duration       1   246068 246072
## <none>               248623 248625
## 
## Step:  AIC=129434
## state ~ backers
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##                  Df Deviance    AIC
## + usd_goal_real  1    68876  68882
## + main_category 14   118530 118562
## + duration       1   127127 127133
## <none>               129430 129434
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=68881.84
## state ~ backers + usd_goal_real
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##                  Df Deviance   AIC
## + main_category 14    62622 62656
## + duration       1    68870 68878
## <none>                68876 68882
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=62655.8
## state ~ backers + usd_goal_real + main_category
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##             Df Deviance   AIC
## + duration  1    62593 62629
## <none>           62622 62656
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=62629.09
## state ~ backers + usd_goal_real + main_category + duration
```

```
stepB <- stepAIC(full, direction= "backward", trace=TRUE)
```

```
## Start:  AIC=62629.09
## state ~ main_category + backers + usd_goal_real + duration
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##                  Df Deviance    AIC
## <none>                62593  62629
## - duration       1    62622  62656
## - main_category 14    68870  68878
## - usd_goal_real  1   116036 116070
## - backers        1   230956 230990
```

```
#Both forward and backward selection methods resulted same in the final model. All the variables will be use
d in this case.
```

Logistic Regression

```
set.seed(224)

glmFit <- glm(state ~ duration + backers + main_category + usd_goal_real, data = ksp.train, family = "binomi
al")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glmFit)
```

```
##
## Call:
## glm(formula = state ~ duration + backers + main_category + usd_goal_real,
##     family = "binomial", data = ksp.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.1416  -0.0808  -0.0029   0.2202   5.0670
##
## Coefficients:
##                              Estimate Std. Error  z value Pr(>|z|)
## (Intercept)                   6.06306    0.07314   82.897  < 2e-16 ***
## duration                     -0.40350    0.07522   -5.364 8.14e-08 ***
## backers                      39.98289    0.22809  175.291  < 2e-16 ***
## main_categoryComics          -1.38314    0.06730  -20.551  < 2e-16 ***
## main_categoryCrafts          -0.86374    0.08223  -10.504  < 2e-16 ***
## main_categoryDance            1.08784    0.09327   11.663  < 2e-16 ***
## main_categoryDesign          -1.14567    0.05250  -21.822  < 2e-16 ***
## main_categoryFashion         -0.40110    0.05772   -6.949 3.67e-12 ***
## main_categoryFilm & Video     0.57417    0.04202   13.665  < 2e-16 ***
## main_categoryFood            -0.47873    0.05240   -9.136  < 2e-16 ***
## main_categoryGames           -2.30496    0.05380  -42.842  < 2e-16 ***
## main_categoryJournalism      -0.25725    0.11464   -2.244   0.0248 *
## main_categoryMusic            0.37920    0.04263    8.894  < 2e-16 ***
## main_categoryPhotography     -0.14538    0.07003   -2.076   0.0379 *
## main_categoryPublishing      -0.44776    0.04725   -9.476  < 2e-16 ***
## main_categoryTechnology      -0.67653    0.05909  -11.450  < 2e-16 ***
## main_categoryTheater          0.91318    0.06546   13.951  < 2e-16 ***
## usd_goal_real               -37.45974    0.25005 -149.808  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 248623  on 182951  degrees of freedom
## Residual deviance:  62593  on 182934  degrees of freedom
## AIC: 62629
##
## Number of Fisher Scoring iterations: 8
```

```
glm.predicted.train <- predict(glmFit, ksp.train, type='response')
glm.predicted_1.train <- ifelse(glm.predicted.train >=0.5, 'successful','failed')
glm.predicted_1.train <- as.factor(glm.predicted_1.train)
glm.results.train <-confusionMatrix(ksp.train$state, glm.predicted_1.train)
glm.results.train
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      99251       7317
##   successful   5327      71057
##
##                Accuracy : 0.9309
##                  95% CI : (0.9297, 0.932)
##     No Information Rate : 0.5716
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8584
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9491
##             Specificity : 0.9066
##          Pos Pred Value : 0.9313
##          Neg Pred Value : 0.9303
##              Prevalence : 0.5716
##          Detection Rate : 0.5425
##    Detection Prevalence : 0.5825
##       Balanced Accuracy : 0.9279
##
##        'Positive' Class : failed
##
```

```
precision_glm.train <- glm.results.train$byClass['Pos Pred Value']
precision_glm.train
```

```
## Pos Pred Value
##      0.9313396
```

```
recall_glm.train <- glm.results.train$byClass['Sensitivity']
recall_glm.train
```

```
## Sensitivity
##   0.9490619
```

```
F1_glm.train <- 2*precision_glm.train*recall_glm.train/(precision_glm.train+recall_glm.train)
F1_glm.train
```

```
## Pos Pred Value
##      0.9401173
```

```
glm.predicted <- predict(glmFit, ksp.test, type='response')
glm.predicted_1 <- ifelse(glm.predicted >=0.5, 'successful','failed')
glm.predicted_1 <- as.factor(glm.predicted_1)
glm.results <-confusionMatrix(ksp.test$state, glm.predicted_1)
glm.results
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      42351       3142
##   successful   2346      30569
##
##                Accuracy : 0.93
##                  95% CI : (0.9282, 0.9318)
##     No Information Rate : 0.5701
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8568
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9475
##             Specificity : 0.9068
##          Pos Pred Value : 0.9309
##          Neg Pred Value : 0.9287
##              Prevalence : 0.5701
##          Detection Rate : 0.5401
##    Detection Prevalence : 0.5802
##       Balanced Accuracy : 0.9272
##
##        'Positive' Class : failed
##
```

```
precision_glm <- glm.results$byClass['Pos Pred Value']
precision_glm
```

```
## Pos Pred Value
##      0.9309344
```

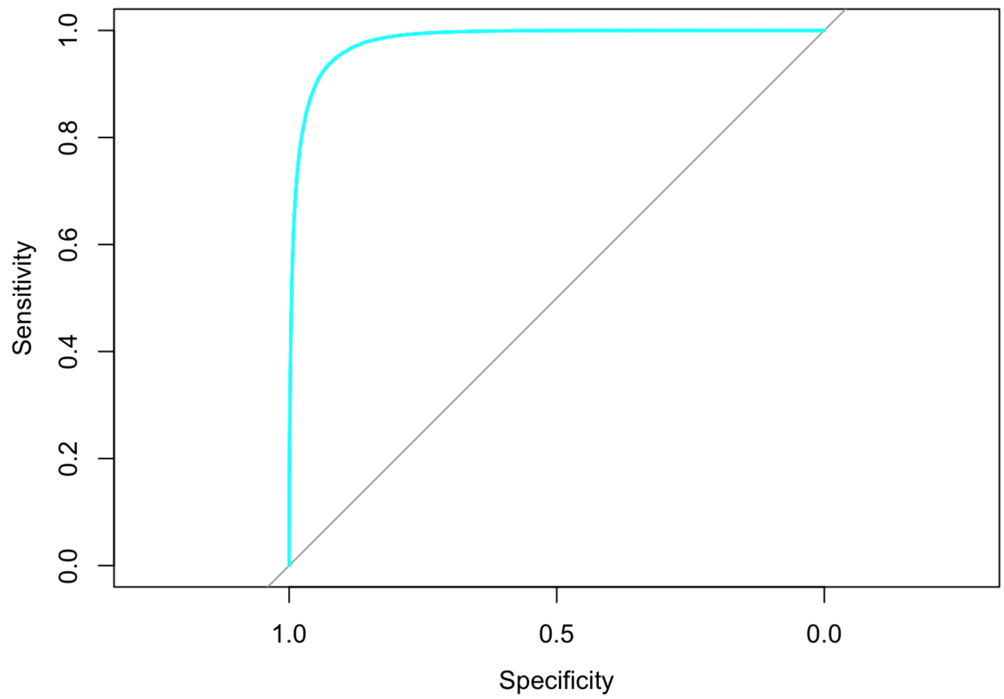
```
recall_glm <- glm.results$byClass['Sensitivity']
recall_glm
```

```
## Sensitivity
##   0.9475133
```

```
F1_glm <- 2*precision_glm*recall_glm/(precision_glm+recall_glm)
F1_glm
```

```
## Pos Pred Value
##      0.9391507
```

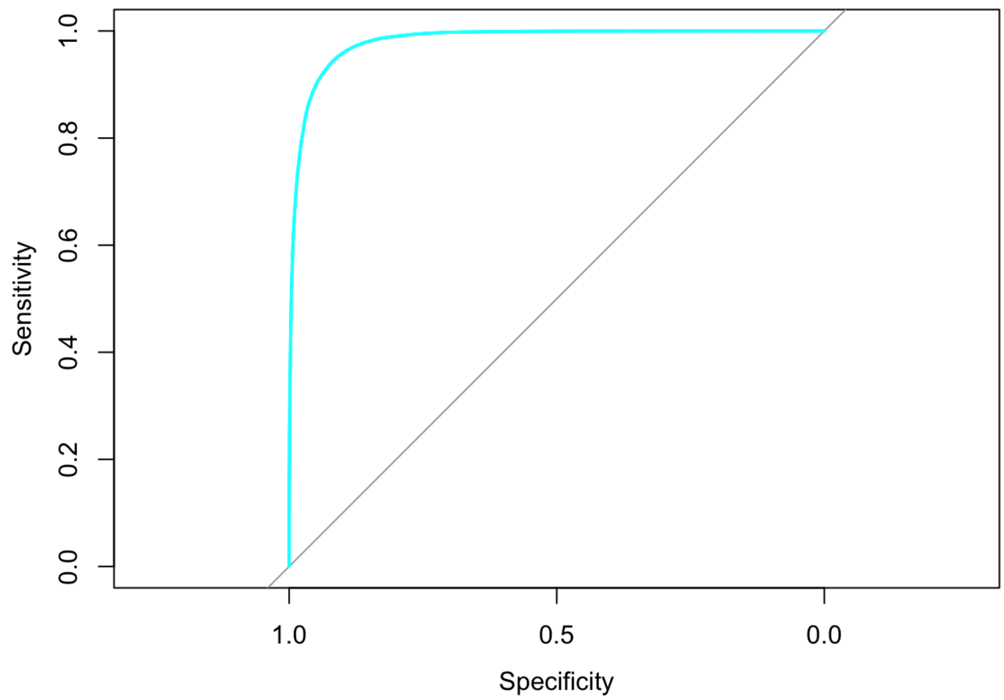
```
rocCurve.glm.train  <-roc(ksp.train$state, glm.predicted.train)
plot(rocCurve.glm.train, type='S', col=c(5))
```




```
auc(rocCurve.glm.train)
```

```
## Area under the curve: 0.9814
```

```
rocCurve.glm  <-roc(ksp.test$state, glm.predicted)
plot(rocCurve.glm, type='S', col=c(5))
```




```
auc(rocCurve.glm)
```

```
## Area under the curve: 0.9815
```

Random Forrest

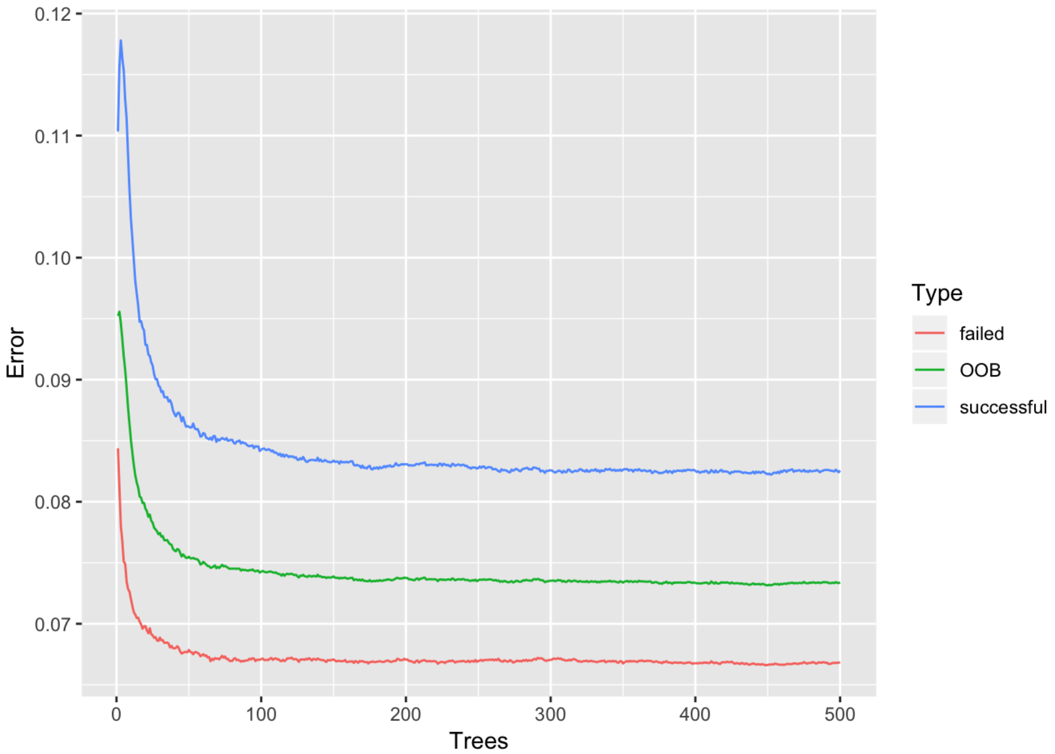
```r
set.seed(224)
rfFit <- randomForest(formula = state~., data= ksp.train, importance=TRUE)
```

```r
print(rfFit)
```

```
## 
## Call:
##  randomForest(formula = state ~ ., data = ksp.train, importance = TRUE) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
## 
##         OOB estimate of  error rate: 7.34%
## Confusion matrix:
##            failed successful class.error
## failed      99447       7121  0.06682118
## successful   6306      70078  0.08255656
```

```r
#The randomforest has a random paremeter as 500 number of trees and 2 variables to split. Let's test which i
s the optimal number of trees and variables to split for my dataset.

oob.error.data <- data.frame(
  Trees=rep(1:nrow(rfFit$err.rate), times=3),
  Type=rep(c("OOB","successful", "failed"), each=nrow(rfFit$err.rate)),
  Error=c(rfFit$err.rate[,"OOB"],
         rfFit$err.rate[,"successful"],
         rfFit$err.rate[,"failed"])
)
```

```r
ggplot(data=oob.error.data, aes(x=Trees, y=Error)) + geom_line(aes(color=Type))
```




```r
oob.values <- vector(length=4)
for(i in 1:4){
  temp.model <- randomForest(state~., data=ksp.train, mtry=i, ntree= 100)
  oob.values[i] <- temp.model$err.rate[nrow(temp.model$err.rate),1]
}
oob.values
```

```
## [1] 0.07014955 0.07410687 0.07750120 0.07851240
```

```r
set.seed(224)
rfFit2 <- randomForest(state~., data=ksp.train, ntree=100 , mtry =1, importance=TRUE)
```

```r
predict.rf.train <- predict(rfFit, ksp.train)
rf.prob.train <- predict(rfFit, ksp.train, type='prob')
confusionMatrix(ksp.train$state, predict.rf.train)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed     105166       1402
##   successful   1661      74723
##
##                Accuracy : 0.9833
##                  95% CI : (0.9827, 0.9838)
##     No Information Rate : 0.5839
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9656
##  Mcnemar's Test P-Value : 3.136e-06
##
##             Sensitivity : 0.9845
##             Specificity : 0.9816
##          Pos Pred Value : 0.9868
##          Neg Pred Value : 0.9783
##              Prevalence : 0.5839
##          Detection Rate : 0.5748
##    Detection Prevalence : 0.5825
##       Balanced Accuracy : 0.9830
##
##        'Positive' Class : failed
##
```

```r
predict.rf.train2 <- predict(rfFit2, ksp.train)
prob.rf.train2 <- predict(rfFit2, ksp.train, type='prob')
results.rf <- confusionMatrix(ksp.train$state, predict.rf.train2)
results.rf
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed     101517       5051
##   successful   3796      72588
##
##                Accuracy : 0.9516
##                  95% CI : (0.9507, 0.9526)
##     No Information Rate : 0.5756
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9008
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9640
##             Specificity : 0.9349
##          Pos Pred Value : 0.9526
##          Neg Pred Value : 0.9503
##              Prevalence : 0.5756
##          Detection Rate : 0.5549
##    Detection Prevalence : 0.5825
##       Balanced Accuracy : 0.9494
##
##        'Positive' Class : failed
##
```

```r
precision_rf <- results.rf$byClass['Pos Pred Value']
precision_rf
```

```
## Pos Pred Value
##       0.952603
```

```
recall_rf <- results.rf$byClass['Sensitivity']
recall_rf
```

```
## Sensitivity
##   0.9639551
```

```
F1_rf <- 2*precision_rf*recall_rf/(precision_rf+recall_rf)
F1_rf
```

```
## Pos Pred Value
##      0.9582454
```

```
predict.rf.test <- predict(rfFit, ksp.test)
prob.rf.test <- predict(rfFit, ksp.test, type='prob')
rf_result.test <- confusionMatrix(ksp.test$state, predict.rf.test)
rf_result.test
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      42493       3000
##   successful   2765      30150
##
##                Accuracy : 0.9265
##                  95% CI : (0.9246, 0.9283)
##     No Information Rate : 0.5772
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8492
##  Mcnemar's Test P-Value : 0.002057
##
##             Sensitivity : 0.9389
##             Specificity : 0.9095
##          Pos Pred Value : 0.9341
##          Neg Pred Value : 0.9160
##              Prevalence : 0.5772
##          Detection Rate : 0.5419
##    Detection Prevalence : 0.5802
##       Balanced Accuracy : 0.9242
##
##        'Positive' Class : failed
##
```

```
precision_rf.test <- rf_result.test$byClass['Pos Pred Value']
precision_rf.test
```

```
## Pos Pred Value
##      0.9340558
```

```
recall_rf.test <- rf_result.test$byClass['Sensitivity']
recall_rf.test
```

```
## Sensitivity
##   0.9389058
```

```
F1_rf.test <- 2*precision_rf.test*recall_rf.test/(precision_rf.test+recall_rf.test)
F1_rf.test
```

```
## Pos Pred Value
##      0.9364745
```

```
rf.predict <- predict(rfFit2, ksp.test)
rf.prob <- predict(rfFit2, ksp.test, type='prob')
rf_result2 <- confusionMatrix(ksp.test$state, rf.predict)
rf_result2
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      42453       3040
##   successful   2378      30537
##
##                Accuracy : 0.9309
##                  95% CI : (0.9291, 0.9327)
##     No Information Rate : 0.5718
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8585
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9470
##             Specificity : 0.9095
##          Pos Pred Value : 0.9332
##          Neg Pred Value : 0.9278
##              Prevalence : 0.5718
##          Detection Rate : 0.5414
##    Detection Prevalence : 0.5802
##       Balanced Accuracy : 0.9282
##
##        'Positive' Class : failed
##
```

```
precision_rf <- rf_result2$byClass['Pos Pred Value']
precision_rf
```

```
## Pos Pred Value
##      0.9331765
```

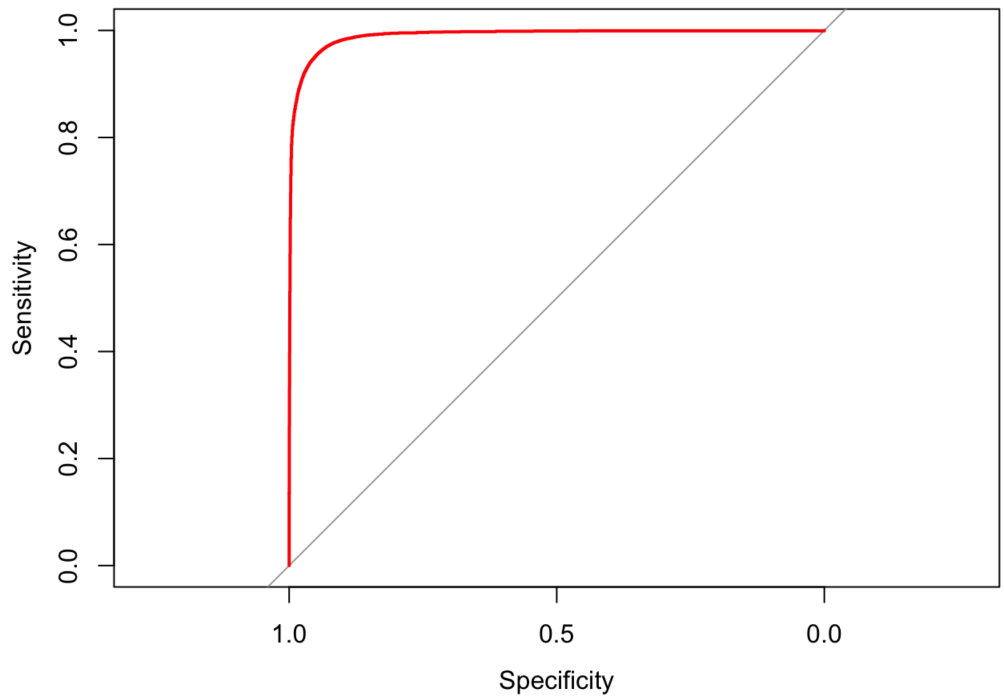
```
recall_rf <- rf_result2$byClass['Sensitivity']
recall_rf
```

```
## Sensitivity
##   0.9469563
```

```
F1_rf <- 2*precision_rf*recall_rf/(precision_rf+recall_rf)
F1_rf
```

```
## Pos Pred Value
##      0.9400159
```

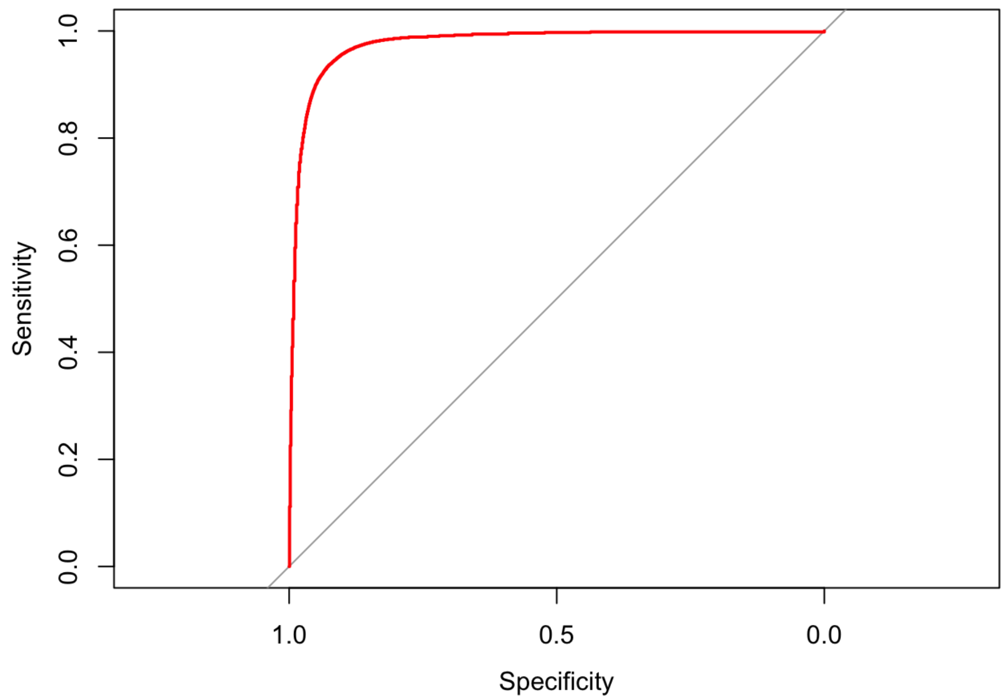
```
rocCurve.rf.train <- roc(ksp.train$state, prob.rf.train2[,2])
plot(rocCurve.rf.train, type= 'S', col=c(2))
```




```
auc(rocCurve.rf.train)
```

```
## Area under the curve: 0.9908
```

```
rocCurve.rf <- roc(ksp.test$state, rf.prob[,2])
plot(rocCurve.rf, type= 'S', col=c(2))
```




```
auc(rocCurve.rf)
```

```
## Area under the curve: 0.9778
```

kNN

```r
ksp.new2 <- data.frame(model.matrix(~.-1, data=kspN))
ksp.new2$statesuccessful <- as.factor(ksp.new2$statesuccessful)

rn_train2 <- sample(nrow(ksp.new2), floor(nrow(ksp.new2)*0.7))
ksp.train2 <- ksp.new2[rn_train2,]
ksp.test2 <- ksp.new2[-rn_train2,]
```

```r
set.seed(224)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knnFit <- train(statesuccessful ~ ., data = ksp.train2, method = "knn", trControl = ctrl, preProcess = c("ce
nter","scale"),tuneLength = 20)
knnFit
```

```
## k-Nearest Neighbors
##
## 182952 samples
##     18 predictor
##      2 classes: '0', '1'
##
## Pre-processing: centered (18), scaled (18)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 164657, 164656, 164656, 164657, 164657, 164657, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9257365  0.8476030
##    7  0.9278226  0.8519484
##    9  0.9290816  0.8545640
##   11  0.9296446  0.8557491
##   13  0.9302057  0.8569180
##   15  0.9302877  0.8571043
##   17  0.9304499  0.8574382
##   19  0.9305629  0.8576765
##   21  0.9305501  0.8576588
##   23  0.9307615  0.8580918
##   25  0.9311313  0.8588647
##   27  0.9311368  0.8588805
##   29  0.9313372  0.8593076
##   31  0.9312643  0.8591680
##   33  0.9312260  0.8590983
##   35  0.9310220  0.8586885
##   37  0.9309710  0.8585965
##   39  0.9309491  0.8585580
##   41  0.9310311  0.8587267
##   43  0.9311313  0.8589401
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 29.
```

```r
knnpredict2 <- predict(knnFit, ksp.train2)
knn.prob2 <- predict(knnFit, ksp.train2, type='prob')
knn_result2 <- confusionMatrix(knnpredict2, ksp.train2$statesuccessful)
knn_result2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 99778  5139
##          1  6762 71273
##
##                Accuracy : 0.935
##                  95% CI : (0.9338, 0.9361)
##     No Information Rate : 0.5823
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8667
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9365
##             Specificity : 0.9327
##          Pos Pred Value : 0.9510
##          Neg Pred Value : 0.9133
##              Prevalence : 0.5823
##          Detection Rate : 0.5454
##    Detection Prevalence : 0.5735
##       Balanced Accuracy : 0.9346
##
##        'Positive' Class : 0
##
```

```
precision_knn <- knn_result2$byClass['Pos Pred Value']
precision_knn
```

```
## Pos Pred Value
##      0.9510184
```

```
recall_knn <- knn_result2$byClass['Sensitivity']
recall_knn
```

```
## Sensitivity
##   0.9365309
```

```
F1_knn <- 2*precision_knn*recall_knn/(precision_knn+recall_knn)
F1_knn
```

```
## Pos Pred Value
##      0.9437191
```

```
knnpredict <- predict(knnFit, ksp.test2)
knn.prob <- predict(knnFit, ksp.test2, type='prob')
knn_result <- confusionMatrix(knnpredict, ksp.test2$statesuccessful)
knn_result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 42366  2410
##          1  3155 30477
##
##                Accuracy : 0.929
##                  95% CI : (0.9272, 0.9308)
##     No Information Rate : 0.5806
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8547
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9307
##             Specificity : 0.9267
##          Pos Pred Value : 0.9462
##          Neg Pred Value : 0.9062
##              Prevalence : 0.5806
##          Detection Rate : 0.5403
##    Detection Prevalence : 0.5711
##       Balanced Accuracy : 0.9287
##
##        'Positive' Class : 0
##
```

```
precision_knn <- knn_result$byClass['Pos Pred Value']
precision_knn
```

```
## Pos Pred Value
##      0.9461765
```

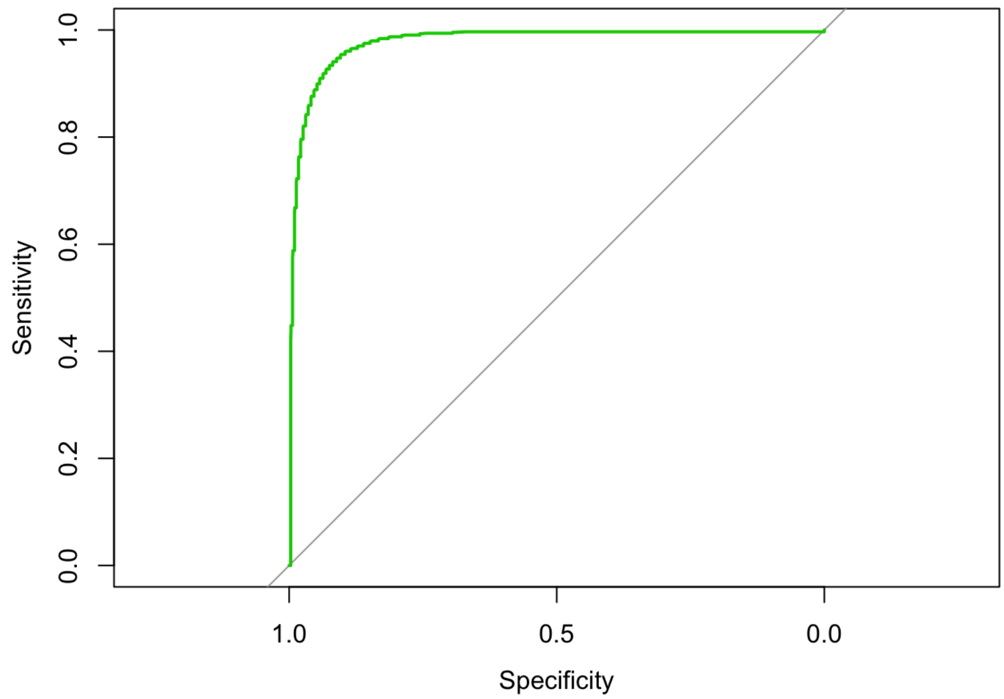
```
recall_knn <- knn_result$byClass['Sensitivity']
recall_knn
```

```
## Sensitivity
##   0.9306913
```

```
F1_knn <- 2*precision_knn*recall_knn/(precision_knn+recall_knn)
F1_knn
```

```
## Pos Pred Value
##        0.93837
```

```
rocCurve.knn <- roc(ksp.test2$state, knn.prob[,2])
plot(rocCurve.knn, type= 'S', col=c(3))
```




```
auc(rocCurve.knn)
```

```
## Area under the curve: 0.9796
```

SVM

```
set.seed(224)
svmFit <- train(state ~ .,
    data = ksp.train,
    method = "svmLinear",
    preProc = c("center", "scale"),
    trControl = trainControl(method = "repeatedcv", repeats = 5, classProbs =  TRUE))
```

```
svm.probs.train <- predict(svmFit, ksp.train, type='prob')
svm.probs <- predict(svmFit, ksp.test, type='prob')
```

```
svm.pred <- predict(svmFit, ksp.train)
svm_result.train <- confusionMatrix(svm.pred, ksp.train$state)

svm_result.train
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      99255       5330
##   successful   7313      71054
##
##                Accuracy : 0.9309
##                  95% CI : (0.9297, 0.9321)
##     No Information Rate : 0.5825
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8584
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9314
##             Specificity : 0.9302
##          Pos Pred Value : 0.9490
##          Neg Pred Value : 0.9067
##              Prevalence : 0.5825
##          Detection Rate : 0.5425
##    Detection Prevalence : 0.5717
##       Balanced Accuracy : 0.9308
##
##        'Positive' Class : failed
##
```

```
precision_svm <- svm_result.train$byClass['Pos Pred Value']
precision_svm
```

```
## Pos Pred Value
##      0.9490367
```

```
recall_svm <- svm_result.train$byClass['Sensitivity']
recall_svm
```

```
## Sensitivity
##   0.9313771
```

```
F1_svm <- 2*precision_svm*recall_svm/(precision_svm+recall_svm)
F1_svm
```

```
## Pos Pred Value
##       0.940124
```

```
svm.pred.test <- predict(svmFit, ksp.test)
svm_result <- confusionMatrix(svm.pred.test, ksp.test$state)

svm_result
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      42360       2344
##   successful   3133      30571
##
##                Accuracy : 0.9301
##                  95% CI : (0.9283, 0.9319)
##     No Information Rate : 0.5802
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8571
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9311
##             Specificity : 0.9288
##          Pos Pred Value : 0.9476
##          Neg Pred Value : 0.9070
##              Prevalence : 0.5802
##          Detection Rate : 0.5403
##    Detection Prevalence : 0.5701
##       Balanced Accuracy : 0.9300
##
##        'Positive' Class : failed
##
```

```
precision_svm <- svm_result$byClass['Pos Pred Value']
precision_svm
```

```
## Pos Pred Value
##      0.9475662
```

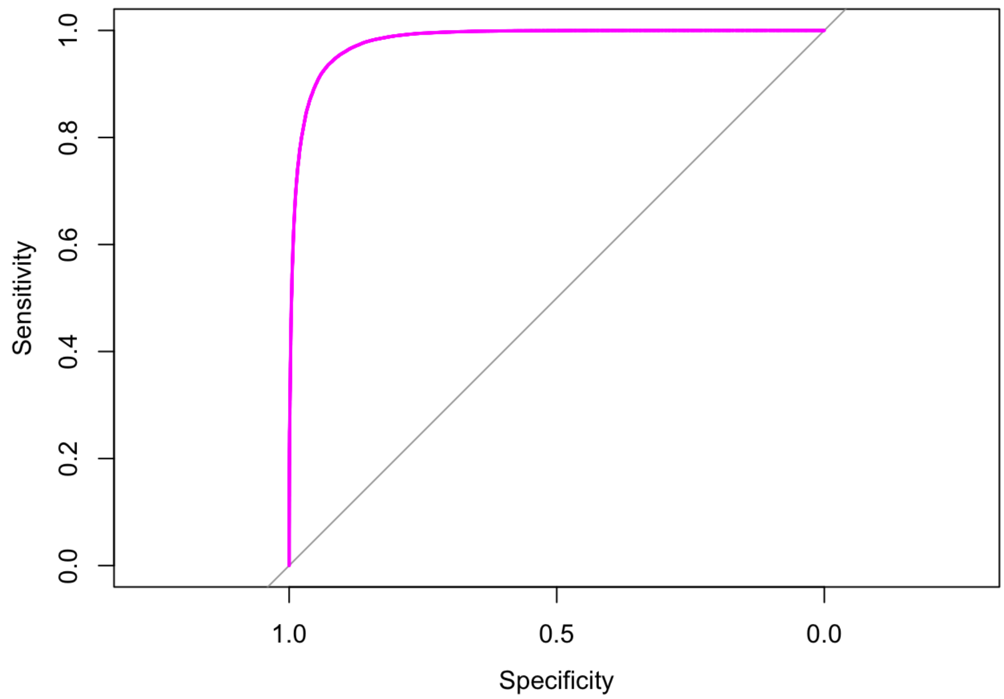
```
recall_svm <- svm_result$byClass['Sensitivity']
recall_svm
```

```
## Sensitivity
##   0.9311323
```

```
F1_svm <- 2*precision_svm*recall_svm/(precision_svm+recall_svm)
F1_svm
```

```
## Pos Pred Value
##      0.9392774
```

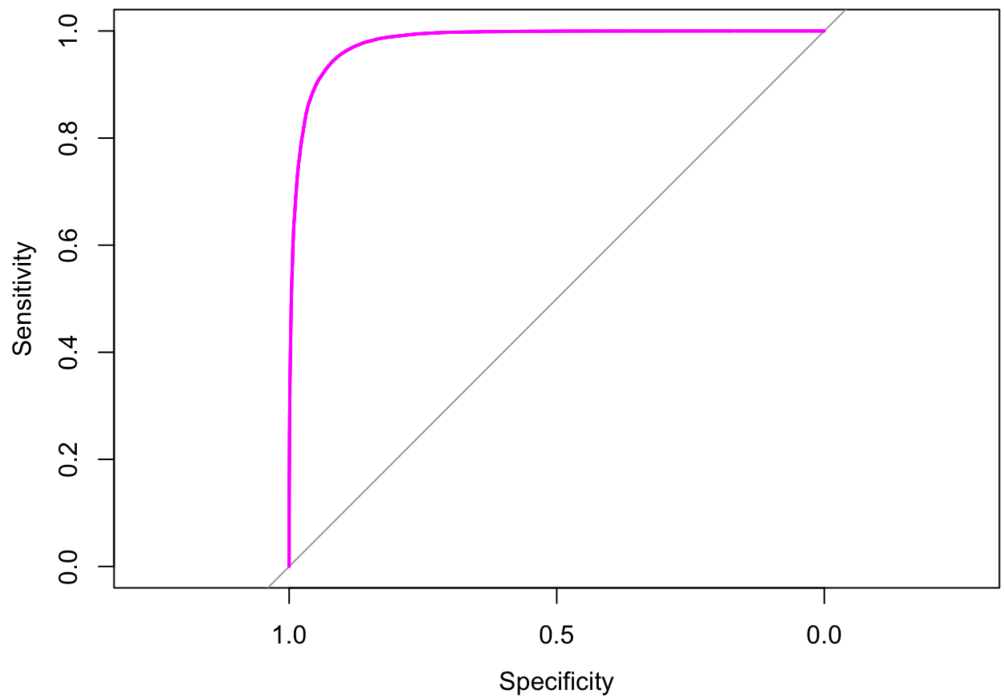
```
rocCurve.svm.train  <-roc(ksp.train$state, svm.probs.train[,2])
plot(rocCurve.svm.train, type='S', col=c(6))
```




```
auc(rocCurve.svm.train)
```

```
## Area under the curve: 0.9814
```

```
rocCurve.svm  <-roc(ksp.test$state, svm.probs[,2])
plot(rocCurve.svm, type='S', col=c(6))
```




```
auc(rocCurve.svm)
```

```
## Area under the curve: 0.9814
```

Ensemble Methods

```r
set.seed(224)
control <- trainControl(method="repeatedcv", number = 10)

train.gbm <- train(state ~ .,
                   data=ksp.train,
                   method="gbm",
                   verbose=F,
                   trControl=control)
train.gbm
```

```
## Stochastic Gradient Boosting
##
## 182952 samples
##      4 predictor
##      2 classes: 'failed', 'successful'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 164658, 164657, 164656, 164657, 164657, 164657, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8923269  0.7808641
##   1                  100      0.9121026  0.8209159
##   1                  150      0.9193723  0.8355034
##   2                   50      0.9179402  0.8326357
##   2                  100      0.9257510  0.8484154
##   2                  150      0.9289650  0.8548115
##   3                   50      0.9232367  0.8432495
##   3                  100      0.9290305  0.8548747
##   3                  150      0.9315449  0.8599058
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
gbm.classTrain <-  predict(train.gbm, ksp.train)
gbm_result.train <- confusionMatrix(ksp.train$state, gbm.classTrain)
gbm_result.train
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      99097       7471
##   successful   5027      71357
##
##                Accuracy : 0.9317
##                  95% CI : (0.9305, 0.9328)
##     No Information Rate : 0.5691
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8602
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9517
##             Specificity : 0.9052
##          Pos Pred Value : 0.9299
##          Neg Pred Value : 0.9342
##              Prevalence : 0.5691
##          Detection Rate : 0.5417
##    Detection Prevalence : 0.5825
##       Balanced Accuracy : 0.9285
##
##        'Positive' Class : failed
##
```

```r
precision_gbm <- gbm_result.train$byClass['Pos Pred Value']
precision_gbm
```

```
## Pos Pred Value
##      0.9298945
```

```r
recall_gbm <- gbm_result.train$byClass['Sensitivity']
recall_gbm
```

```
## Sensitivity
##    0.951721
```

```r
F1_gbm <- 2*precision_gbm*recall_gbm/(precision_gbm+recall_gbm)
F1_gbm
```

```
## Pos Pred Value
##      0.9406812
```

```r
gbm.classTest <-  predict(train.gbm, ksp.test)
gbm_result <- confusionMatrix(ksp.test$state, gbm.classTest)
gbm_result
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   failed successful
##   failed      42282       3211
##   successful   2205      30710
##
##                Accuracy : 0.9309
##                  95% CI : (0.9291, 0.9327)
##     No Information Rate : 0.5674
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8588
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9504
##             Specificity : 0.9053
##          Pos Pred Value : 0.9294
##          Neg Pred Value : 0.9330
##              Prevalence : 0.5674
##          Detection Rate : 0.5393
##    Detection Prevalence : 0.5802
##       Balanced Accuracy : 0.9279
##
##        'Positive' Class : failed
##
```

```r
precision_gbm <- gbm_result$byClass['Pos Pred Value']
precision_gbm
```

```
## Pos Pred Value
##      0.9294177
```

```r
recall_gbm <- gbm_result$byClass['Sensitivity']
recall_gbm
```

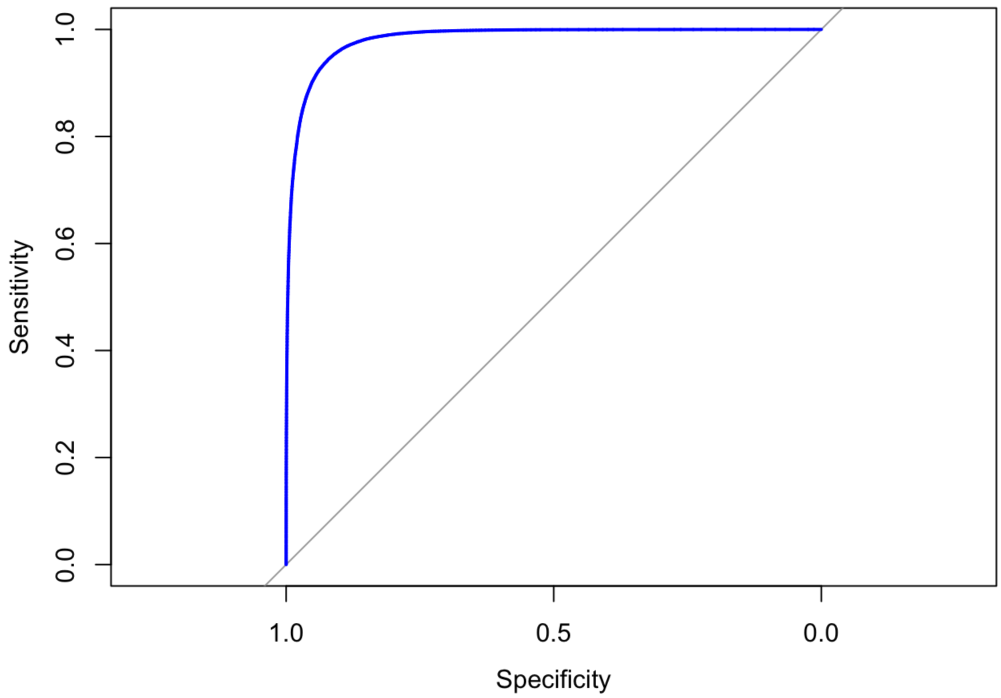
```
## Sensitivity
##    0.950435
```

```r
F1_gbm <- 2*precision_gbm*recall_gbm/(precision_gbm+recall_gbm)
F1_gbm
```

```
## Pos Pred Value
##      0.9398088
```

```
gbm.probs=predict(train.gbm,
                  ksp.test,
                  type="prob")
gbm.probs.train <- predict(train.gbm, ksp.train, type='prob')
```

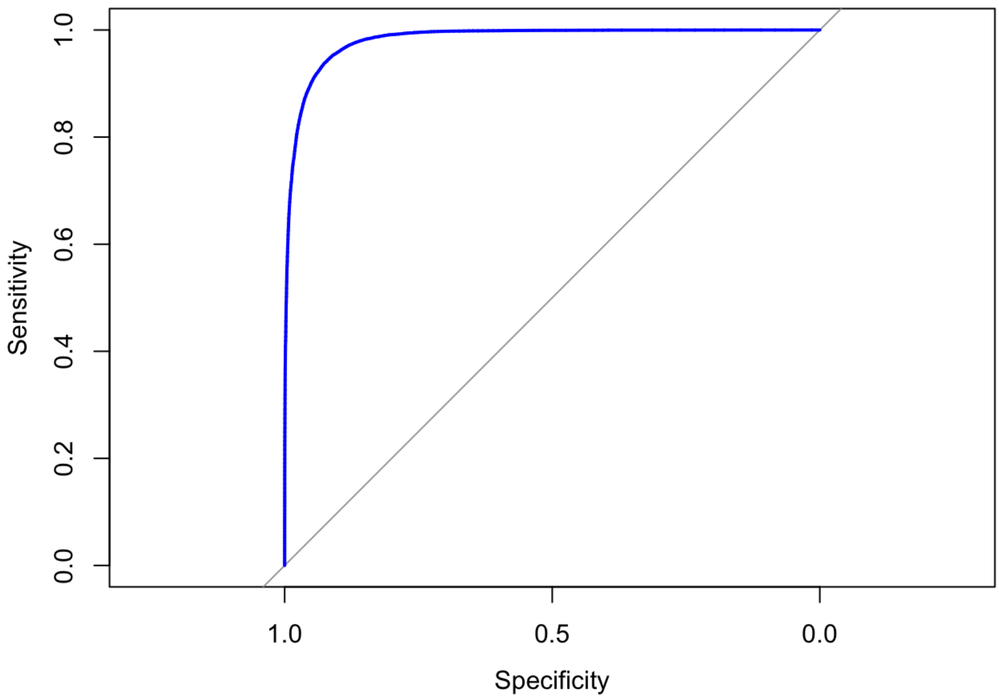
```
rocCurve.gbm.train <- roc(ksp.train$state, gbm.probs.train[,2])
plot(rocCurve.gbm.train, col=c(4))
```




```
auc(rocCurve.gbm.train)
```

```
## Area under the curve: 0.9826
```

```
rocCurve.gbm <- roc(ksp.test$state,gbm.probs[,2])

plot(rocCurve.gbm, col=c(4))
```




```
auc(rocCurve.gbm)
```

```
## Area under the curve: 0.9824
```

```
plot(rocCurve.glm, type="S",main= 'ROC Curve Comparison', col="red")
plot(rocCurve.rf, type="S", add = TRUE, col="green")
plot(rocCurve.knn, type="S",add = TRUE, col="blue")
plot(rocCurve.gbm, type='S',add = TRUE, col="orange")
plot(rocCurve.svm, type='S', add = TRUE, col ='pink')
legend("right", legend=c('GLM ', '  RF  ', 'KNN ', 'GBM ', 'SVM ' ), col=c("red", "green", 'blue','orange',
'pink'), lty=1, cex=0.9)
```