Capstone Project - Predicting Patient No-Shows Using Appointment Data

Derek Samsom

```
library(tidyverse)
library(lubridate)
library(caret)
library(randomForest)
library(GGally)
```

Missed medical appointments are a major problem in the medical industry, resulting in lost revenue and a mismatch between the number of patients and medical staff at any given time. Medical providers can over-book appointments to minimize the lost revenue, but there will still be times that there are more or fewer patients than expected.

This project is a classification problem that will explore the prediction of whether a medical appointment will be missed. By predicting missed appointments, they can be over-booked in a way that reduces missed revenue and increases patient care by reducing the number of times there are more patients than expected.

The scope of this project is limited to predicting whether an appointment will be missed, and does not include the application of this information in an appointment booking system.

Read data and assign to *appointments*

```
appointments <- read_csv("Final_Data.csv")
```

```
## Parsed with column specification:
## cols(
##   kept_status = col_character(),
##   appt_date = col_character(),
##   appt_time = col_time(format = ""),
##   appt_length = col_integer(),
##   date_scheduled = col_character(),
##   patient_age = col_integer(),
##   patient_gender = col_character(),
##   billing_type = col_character(),
##   prior_missed = col_integer(),
##   prior_kept = col_integer(),
##   patient_distance = col_integer(),
##   office_zip = col_character(),
##   provider_specialty = col_character(),
##   remind_call_result = col_character()
## )
```

```
appointments_original <- appointments
zipcodes <- read_csv("zipcodes.csv")
```

```
## Parsed with column specification:
## cols(
##   office_zip = col_character(),
##   county_code = col_character()
## )
```

The raw data, which has been named *appointments*, contains information on 342862 past appointments, sorted by the date and time of appointment. The dependended variable, *kept_status*, shows whether the appintment was be kept or missed.

There is no field that can be used to identify a specific patients in the data set. A patient may have had more than one appointment during the time-period represented in the data, meaning that one individual patient may make up one or multiple observations. If there was a patient ID field, it would allow the data to be grouped by patient and give the option of organizing the data by patient rather than by appointment.

A secondary data set, named *zipcodes*, has information about the county and city sizes. This will be used to see if the location and size of city can help predict whether an appointment will be missed. The county names are converted to a 2-letter code for confidentiality.

Data Summary and Structure

```
summary(appointments)
```

```
##  kept_status         appt_date          appt_time         appt_length
##  Length:342862      Length:342862      Length:342862     Min.   : 10
##  Class :character   Class :character   Class1:hms        1st Qu.: 60
##  Mode  :character   Mode  :character   Class2:difftime   Median : 60
##                                        Mode  :numeric    Mean   : 57
##                                                          3rd Qu.: 60
##                                                          Max.   :600
##
##  date_scheduled      patient_age     patient_gender     billing_type
##  Length:342862      Min.   :  0.00   Length:342862      Length:342862
##  Class :character   1st Qu.: 17.00   Class :character   Class :character
##  Mode  :character   Median : 34.00   Mode  :character   Mode  :character
##                     Mean   : 35.56
##                     3rd Qu.: 54.00
##                     Max.   :264.00
##
##   prior_missed       prior_kept     patient_distance  office_zip
##  Min.   :  0.000   Min.   :  0.00   Min.   :   0.0   Length:342862
##  1st Qu.:  1.000   1st Qu.:  2.00   1st Qu.:   0.0   Class :character
##  Median :  2.000   Median :  6.00   Median :   3.0   Mode  :character
##  Mean   :  2.451   Mean   :  8.02   Mean   :  10.8
##  3rd Qu.:  3.000   3rd Qu.: 11.00   3rd Qu.:   9.0
##  Max.   :117.000   Max.   :676.00   Max.   :2688.0
##                                     NA's   :974
##  provider_specialty remind_call_result
##  Length:342862      Length:342862
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
str(appointments, give.attr = FALSE)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    342862 obs. of  14 variables:
##  $ kept_status       : chr  "Kept" "Kept" "Kept" "Kept" ...
##  $ appt_date         : chr  "9/1/16" "9/1/16" "9/1/16" "9/1/16" ...
##  $ appt_time         :Classes 'hms', 'difftime'  atomic [1:342862] 19800 28800 28800 28800 28800 288
##  $ appt_length       : int  90 60 120 60 60 60 60 60 60 90 ...
##  $ date_scheduled    : chr  "8/1/16" "1/18/16" "2/3/16" "6/8/16" ...
##  $ patient_age       : int  7 75 31 45 49 71 49 38 36 13 ...
##  $ patient_gender    : chr  "Male" "Female" "Male" "Male" ...
##  $ billing_type      : chr  "DMAP" "Commercial" "DMAP" "DMAP" ...
##  $ prior_missed      : int  1 2 1 6 5 6 8 0 2 3 ...
##  $ prior_kept        : int  3 5 5 15 6 6 20 0 5 12 ...
##  $ patient_distance  : int  41 29 5 5 0 5 0 539 0 4 ...
##  $ office_zip        : chr  "AP" "BL" "BL" "BL" ...
##  $ provider_specialty: chr  "A" "A" "A" "B" ...
##  $ remind_call_result: chr  "Left Message" "Answered - Confirmed" "Left Message" "Answered - No Respo
```

```r
head(appointments[,1:5])
```

```
## # A tibble: 6 x 5
##   kept_status appt_date appt_time appt_length date_scheduled
##   <chr>       <chr>     <time>          <int> <chr>
## 1 Kept        9/1/16    05:30              90 8/1/16
## 2 Kept        9/1/16    08:00              60 1/18/16
## 3 Kept        9/1/16    08:00             120 2/3/16
## 4 Kept        9/1/16    08:00              60 6/8/16
## 5 Missed      9/1/16    08:00              60 6/28/16
## 6 Kept        9/1/16    08:00              60 7/12/16
```

```r
head(appointments[,6:10])
```

```
## # A tibble: 6 x 5
##   patient_age patient_gender billing_type prior_missed prior_kept
##         <int> <chr>          <chr>               <int>      <int>
## 1           7 Male           DMAP                    1          3
## 2          75 Female         Commercial              2          5
## 3          31 Male           DMAP                    1          5
## 4          45 Male           DMAP                    6         15
## 5          49 Male           Commercial              5          6
## 6          71 Male           DMAP                    6          6
```

```r
head(appointments[,11:14])
```

```
## # A tibble: 6 x 4
##   patient_distance office_zip provider_specialty remind_call_result
##              <int> <chr>      <chr>              <chr>
## 1               41 AP         A                  Left Message
## 2               29 BL         A                  Answered - Confirmed
## 3                5 BL         A                  Left Message
## 4                5 BL         B                  Answered - No Response
## 5                0 BL         B                  Answered - No Response
## 6                5 BL         A                  Answered - Confirmed
```

Data Dictionary

```r
variable_description <- c(
    "Dependent variable: kept or missed",
    "Appointment date",
    "Appointment time",
    "Appointment length in minutes",
    "Date appointment was scheduled",
    "Patient age",
    "Patient gender",
    "Billing type",
    "Number of prior missed appointments",
    "Number of prior kept appointments",
    "Patient distance from office in miles",
    "Office Zip Code - Anonymized",
    "Provider primary specialty code",
    "Reminder Call result")
variable <- colnames(appointments)

as_data_frame(cbind(c(1:length(variable)), variable, variable_description))
```

```
## # A tibble: 14 x 3
##    V1    variable           variable_description
##    <chr> <chr>              <chr>
##  1 1     kept_status        Dependent variable: kept or missed
##  2 2     appt_date          Appointment date
##  3 3     appt_time          Appointment time
##  4 4     appt_length        Appointment length in minutes
##  5 5     date_scheduled     Date appointment was scheduled
##  6 6     patient_age        Patient age
##  7 7     patient_gender     Patient gender
##  8 8     billing_type       Billing type
##  9 9     prior_missed       Number of prior missed appointments
## 10 10    prior_kept         Number of prior kept appointments
## 11 11    patient_distance   Patient distance from office in miles
## 12 12    office_zip         Office Zip Code - Anonymized
## 13 13    provider_specialty Provider primary specialty code
## 14 14    remind_call_result Reminder Call result
```

Will combine the appointment time and date into one variable, appt_datetime.

```
appointments <- appointments %>%
    mutate(appt_datetime = lubridate::mdy_hms(paste(appt_date, appt_time)))

appointments$date_scheduled <- as.POSIXct(
    appointments$date_scheduled, format = "%m/%d/%y")
```

Calculating percent of missed appointments overall. Will first create a logical variable *missed*, where 1 represents a missed appointment and 0 represents a kept appointment.

```
appointments <- appointments %>%
    mutate(missed = ifelse(appointments$kept_status == "Missed", 1, 0))
missed_rate <- mean(appointments$missed)
missed_rate
```

```
## [1] 0.1592944
```

15.929441 % of the total appointments are missed.

Data Exploration

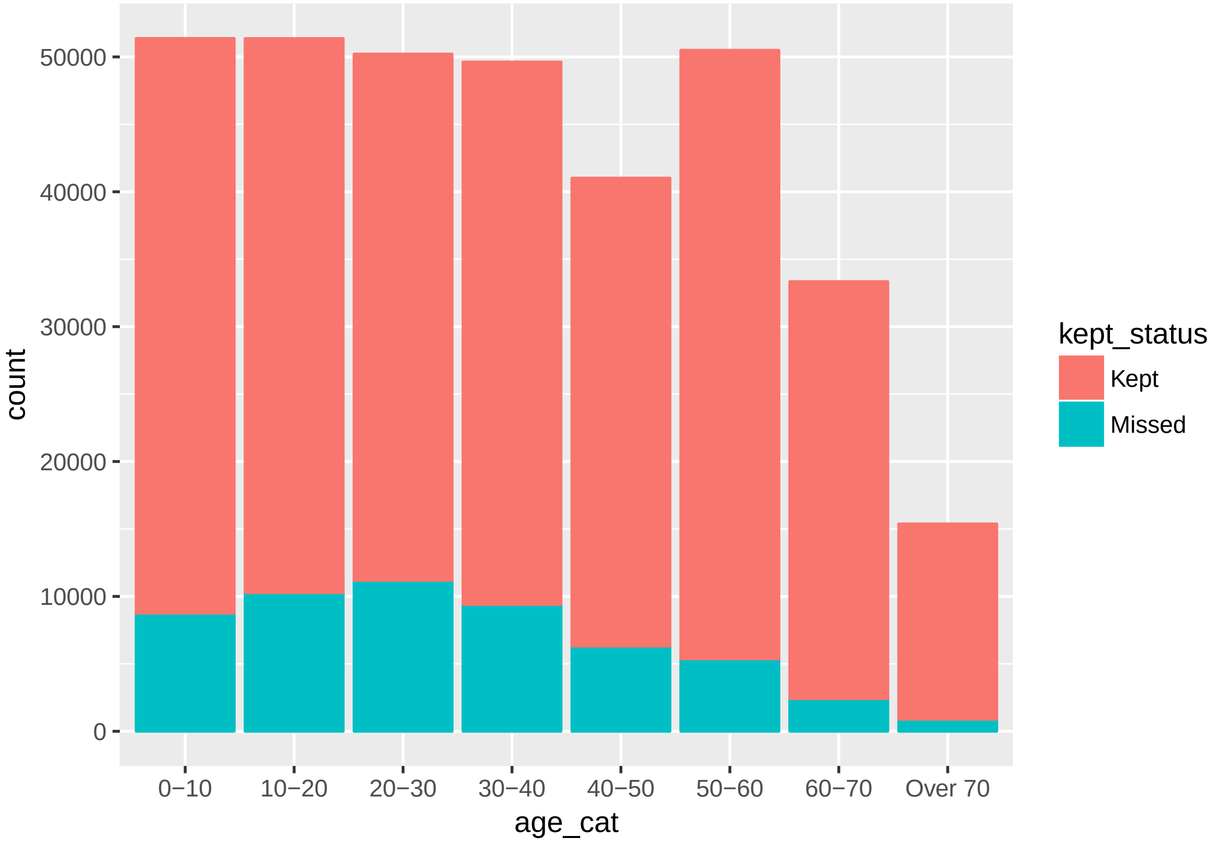
```
# Check for NAs
map_dbl(appointments, ~sum(is.na(.)))
```

```
##        kept_status          appt_date          appt_time
##                  0                  0                  0
##        appt_length     date_scheduled        patient_age
##                  0                  0                  0
##     patient_gender       billing_type       prior_missed
##                  0                  0                  0
##         prior_kept   patient_distance         office_zip
##                  0                974                  0
## provider_specialty remind_call_result      appt_datetime
##                  0                  0                  0
##             missed
##                  0
```

Variable *patient_distance* has 974 NA values.

patient_age

```
age_labels <- c("0-10", "10-20","20-30", "30-40", "40-50", "50-60", "60-70",
                "Over 70")
age_breaks <- c(-1, 10, 20, 30, 40, 50, 60, 70, 111)

appointments <- appointments %>%
    filter(patient_age <= 110) %>%
    mutate(
        age_cat = cut(patient_age, breaks = age_breaks, labels = age_labels))

age_labels <- c("0-10", "10-20","20-30", "30-40", "40-50", "50-60", "60-70",
                "Over 70")
age_breaks <- c(-1, 10, 20, 30, 40, 50, 60, 70, 111)

ggplot(
    appointments,
    aes(x = age_cat, color = kept_status, fill = kept_status)
) +
    stat_count()
```



The data contained a small number of observations that were higher than plausible. Therefore, the observations greater than age 110 were removed from the data.

Missed appointments are highest with young adults, and decrease with older and younger patients.

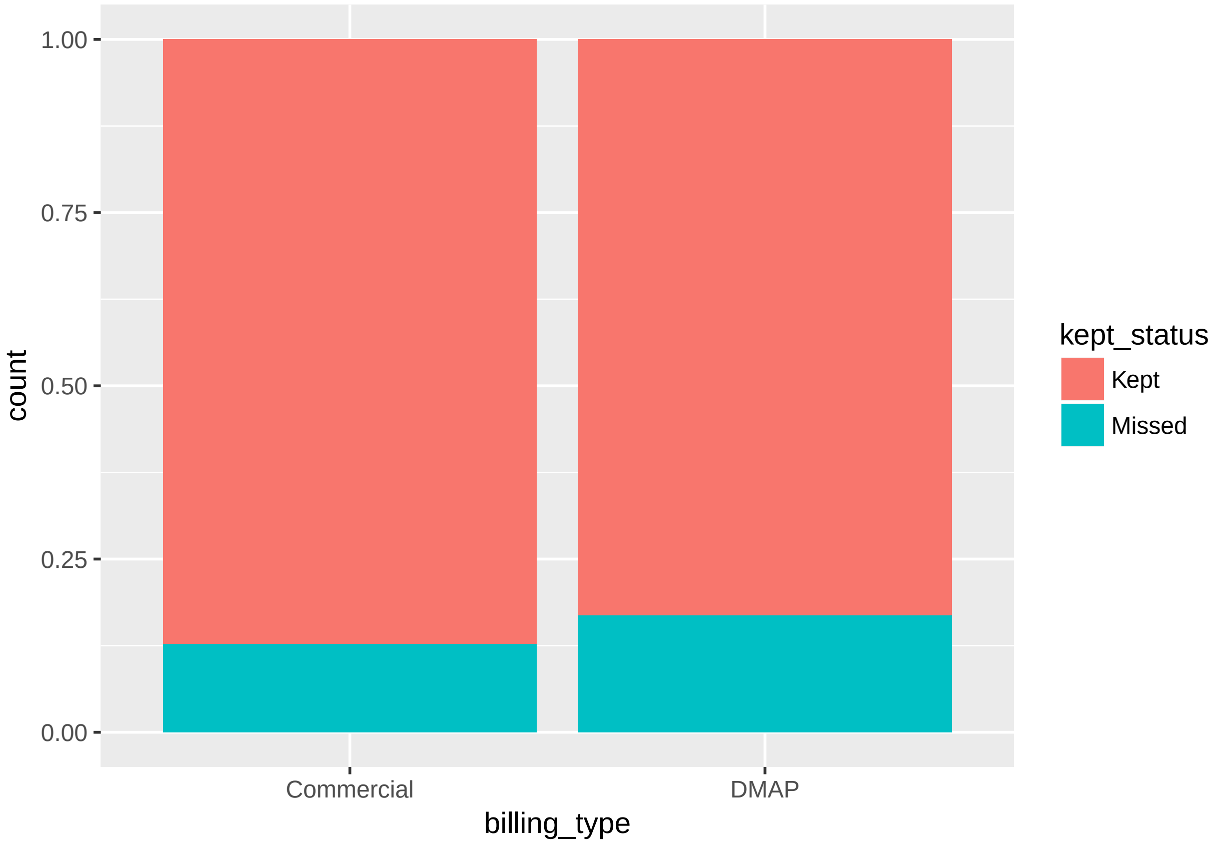
billing_type

```
table(appointments$billing_type)
```

```
##
##      Commercial           DMAP To Be Assigned
##           78282         264500              1
```

There is only one observation of “To Be Assigned”, therefore it will be removed from the data.

```
appointments <- subset(appointments, billing_type != "To Be Assigned")

ggplot(
    appointments,
    aes(x = billing_type, fill = kept_status)
) +
    geom_bar(position = "fill")
```



There is a minor difference between billing types. DMAP has a higher proportion of missed appointments

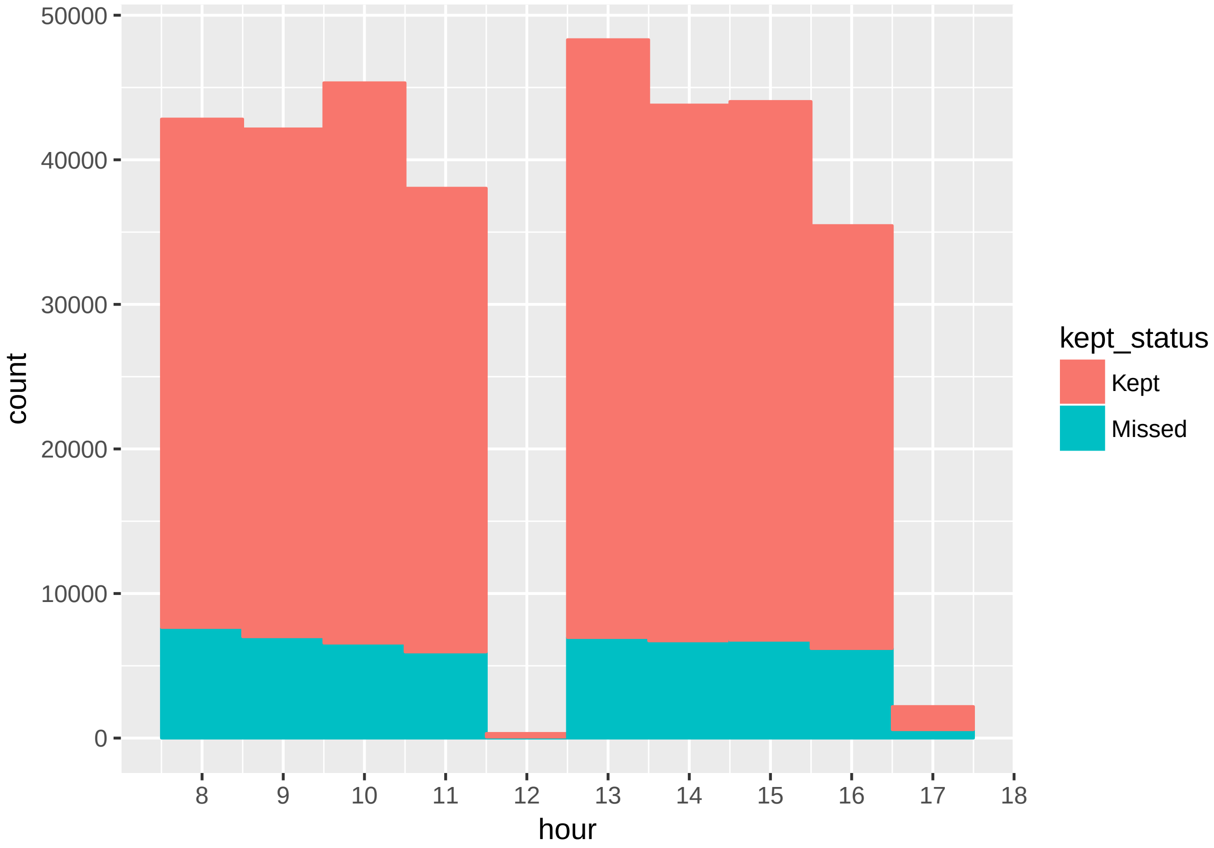
appt_datetime

Creating new *hour* variable and plot by hour

```
appointments <- appointments %>%
    mutate(hour = lubridate::hour(appointments$appt_datetime))

table(appointments$hour)
```

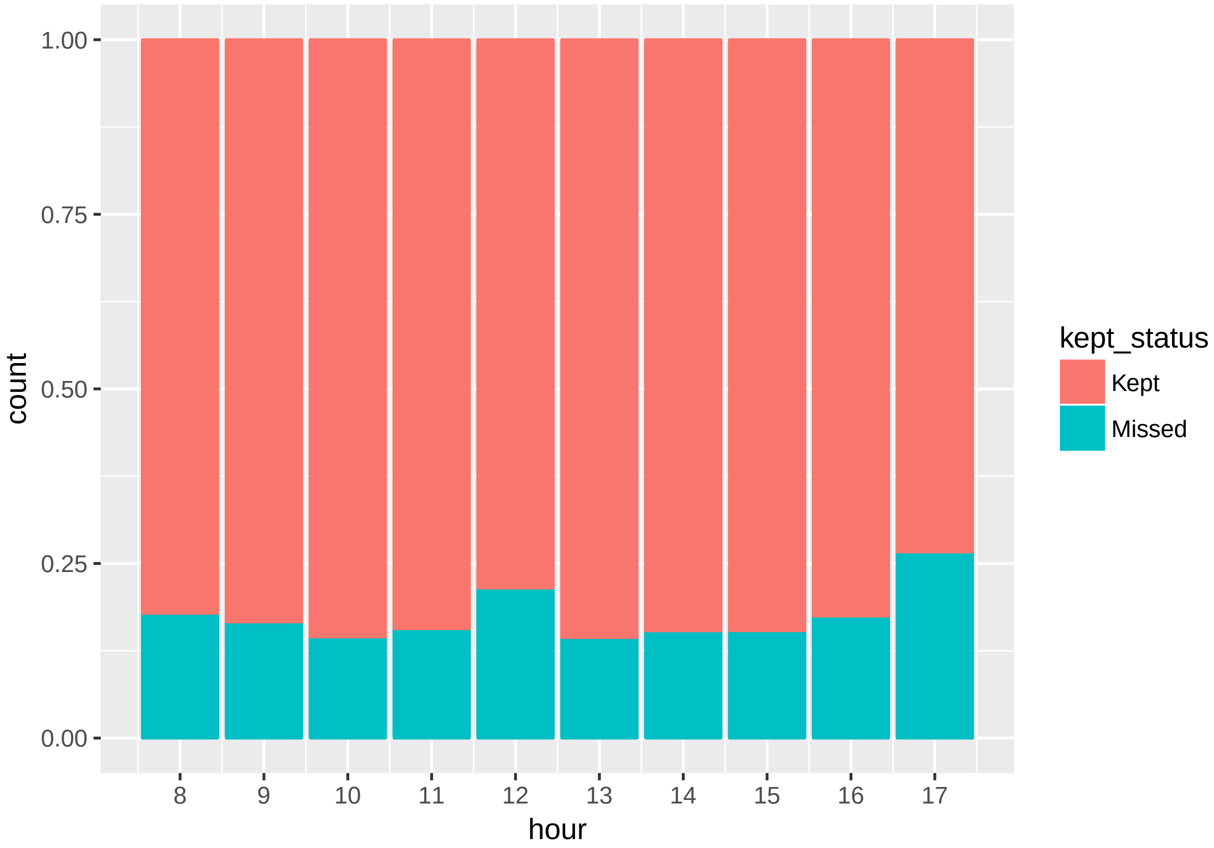
```
##
##     0     5     6     7     8     9    10    11    12    13    14    15
##     7    24    25    98 42816 42133 45326 38033   321 48307 43787 44033
##    16    17    18    19    20    21
## 35449  2180   205    33     3     2
```

```
appointments_hour <- appointments %>%
select(kept_status, hour) %>%
filter(hour > 7 & hour < 18)

ggplot(
    appointments_hour,
    aes(x = hour, col = kept_status, fill = kept_status)
) +
    geom_histogram(binwidth = 1) +
    scale_x_continuous(breaks = seq(8, 18, 1))
```



Most appointments are scheduled between 8:00 AM and 5:00 PM.

```
ggplot(
    appointments_hour,
    aes(x = hour, col = kept_status, fill = kept_status)
) +
    geom_bar(position = "fill") +
    scale_x_continuous(breaks = seq(8, 18, 1))
```



Proportionally more appointments are missed at the beginning and end of the typical scheduling hours, and during the few noon appointments.

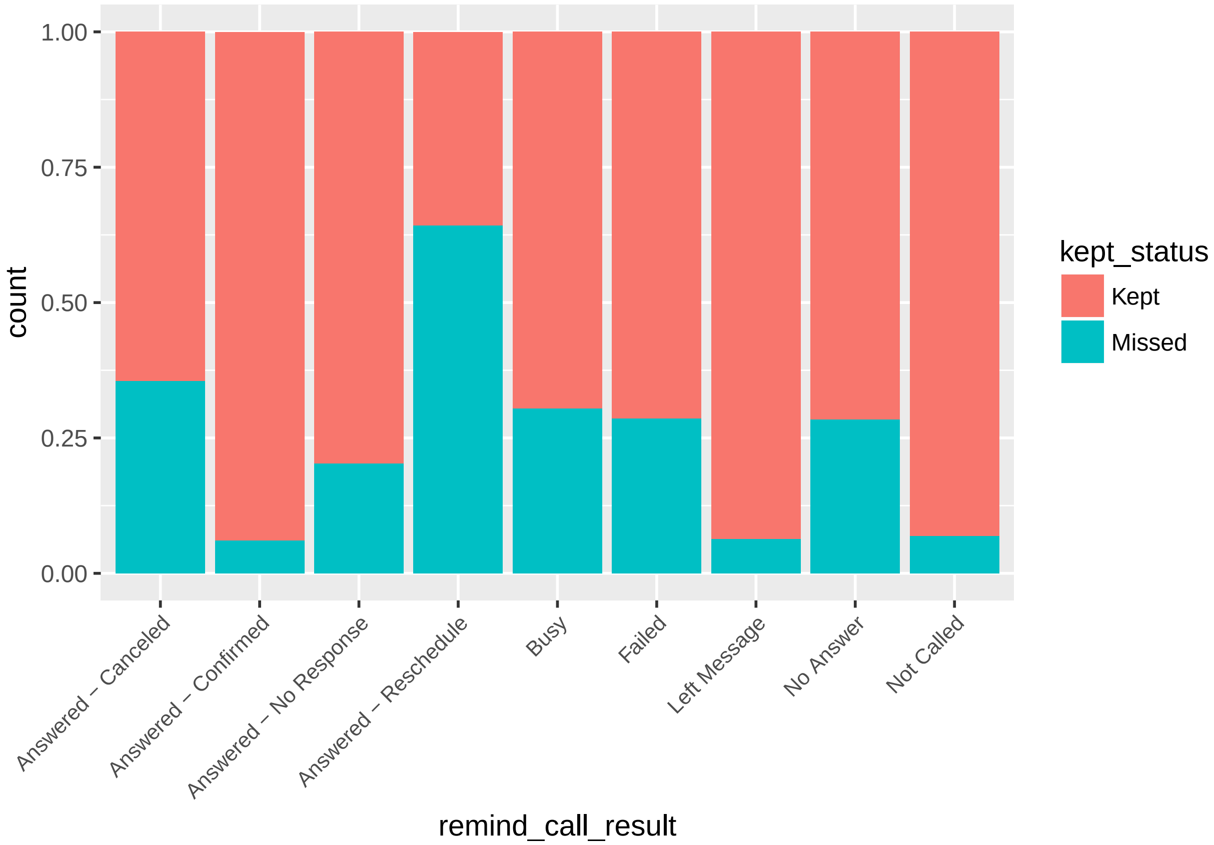
remind_call_result

```
table(appointments$remind_call_result)
```

```
##
##    Answered - Canceled   Answered - Confirmed Answered - No Response
##                    152                  49108                 180869
##  Answered - Reschedule                   Busy                 Failed
##                   1369                   1104                  27944
##           Left Message              No Answer             Not Called
##                  18430                    377                  63429
```

Low counts of “Answered - Cancelled”, “Answered - Reschedule”, “Busy”, and “No Answer”

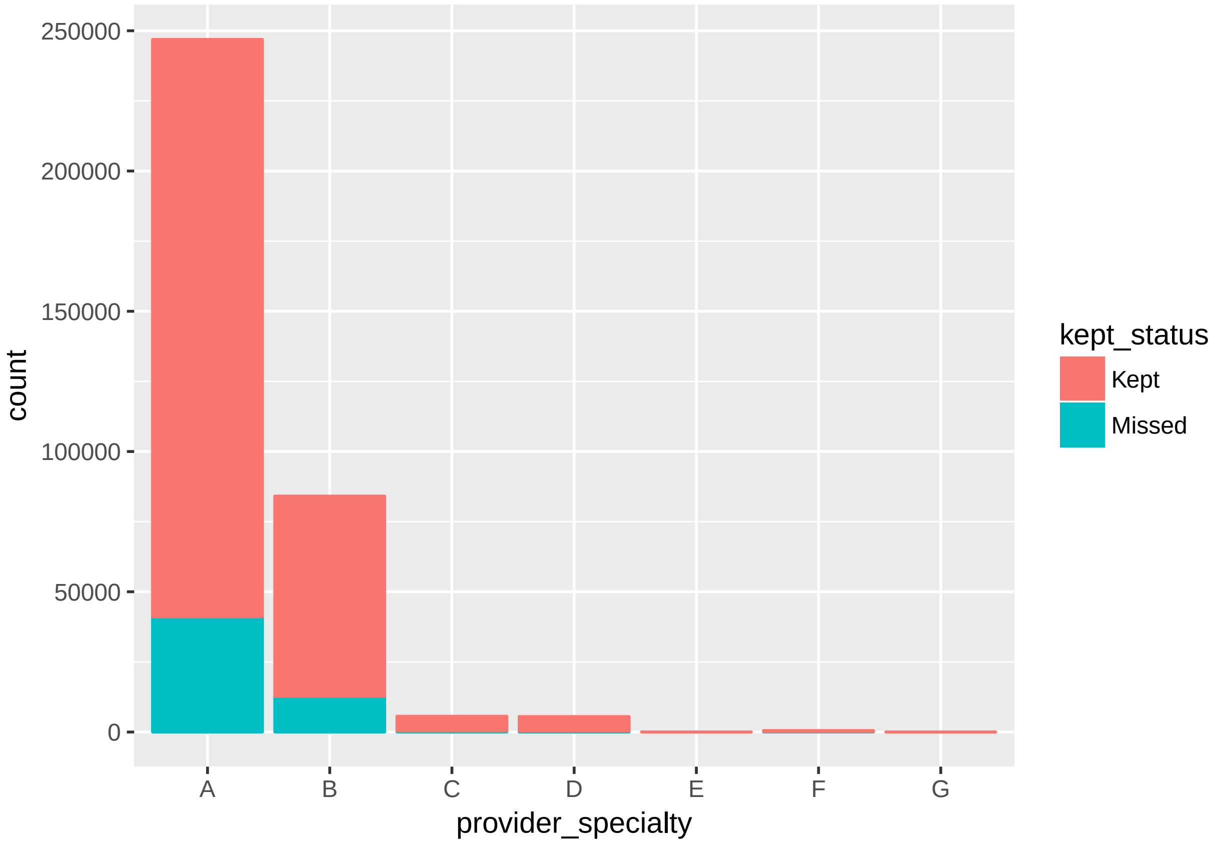
```
ggplot(
    appointments,
    aes(x = remind_call_result, fill = kept_status)
) +
    geom_bar(position = "fill") +
    theme(axis.text.x = element_text(size = 8, angle = 45,hjust = 1, vjust = 1))
```



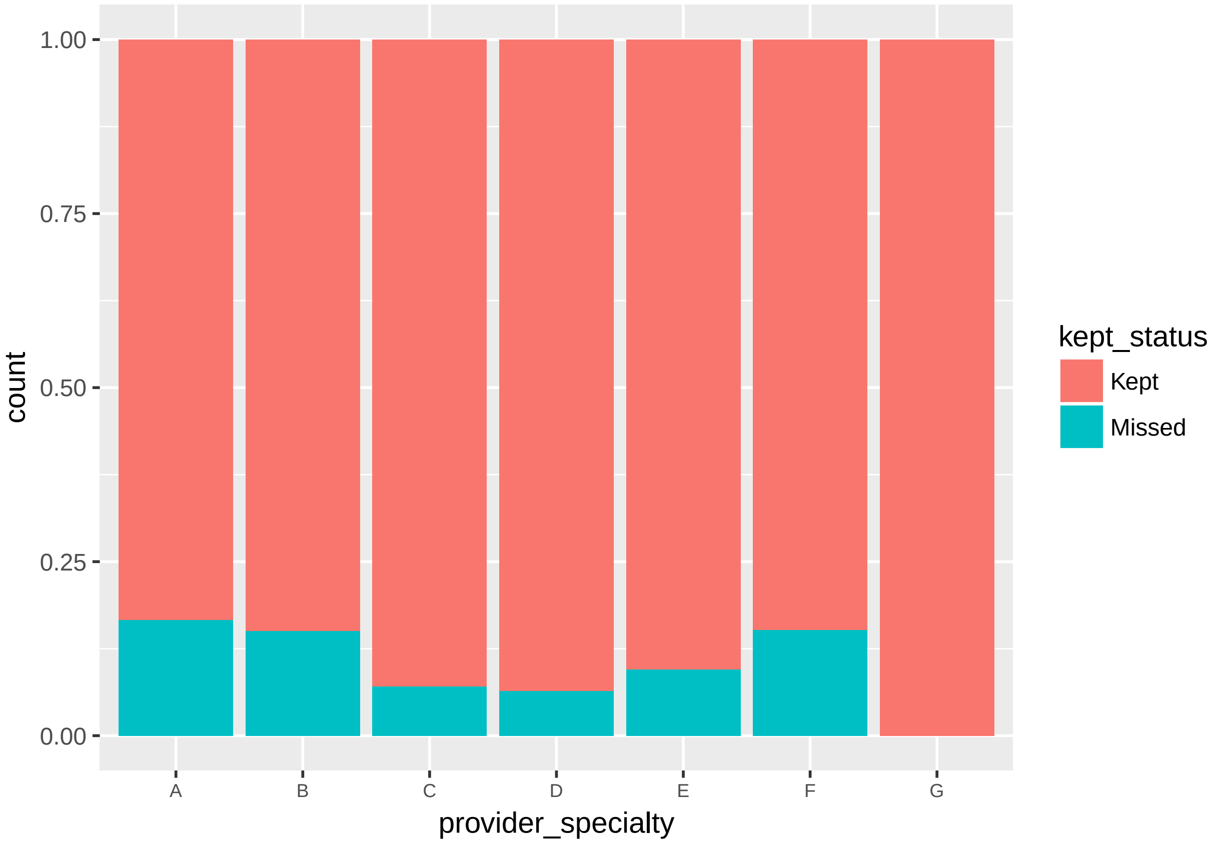
~65% of appointments with “Answered - Cancelled” and ~35% with “Answered-Reschedule” still kept their appointments, however, very few observations in these categories.

provider_specialty

```
ggplot(
    appointments,
    aes(x = provider_specialty, col = kept_status, fill = kept_status)
) +
    stat_count()
```



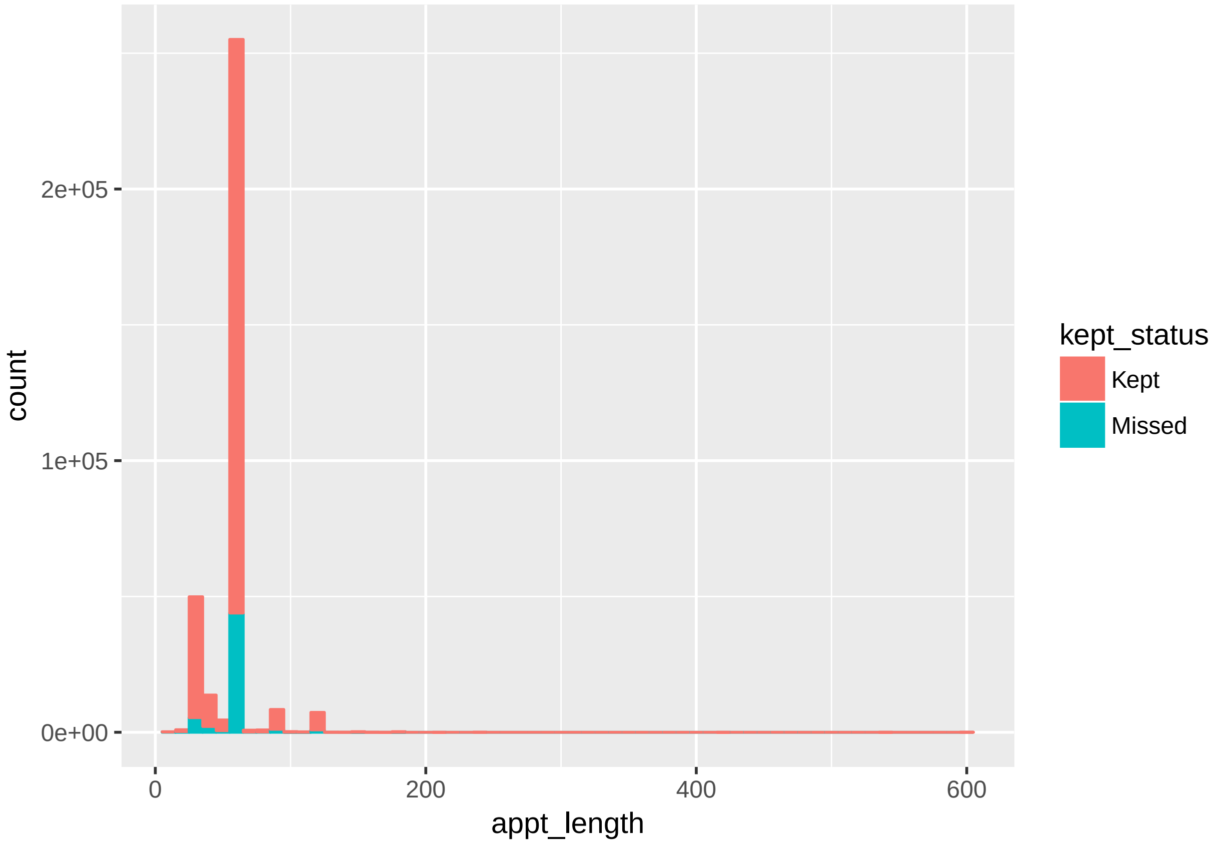
```
ggplot(
    appointments,
    aes(x = provider_specialty, fill = kept_status)
) +
    geom_bar(position = "fill") +
    theme(axis.text.x = element_text(size = 7))
```



C, D, and E provider specialties have lower proportion of missed appointments,

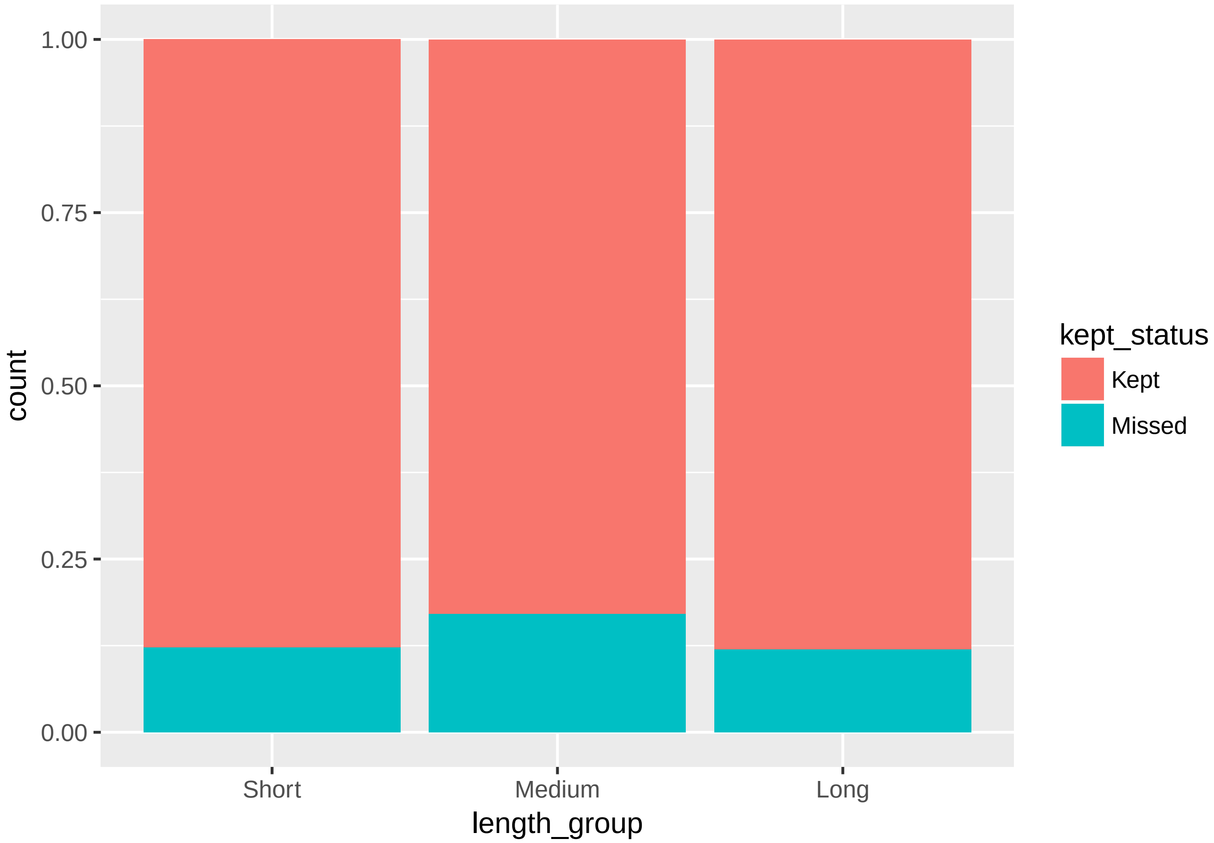
appt_length

```
ggplot(
    data = appointments,
    mapping = aes(x = appt_length, col = kept_status, fill = kept_status)
) +
    geom_histogram(binwidth = 10)
```



Most Appointments are 60 minutes long. 30-minute appointments are next most popular.

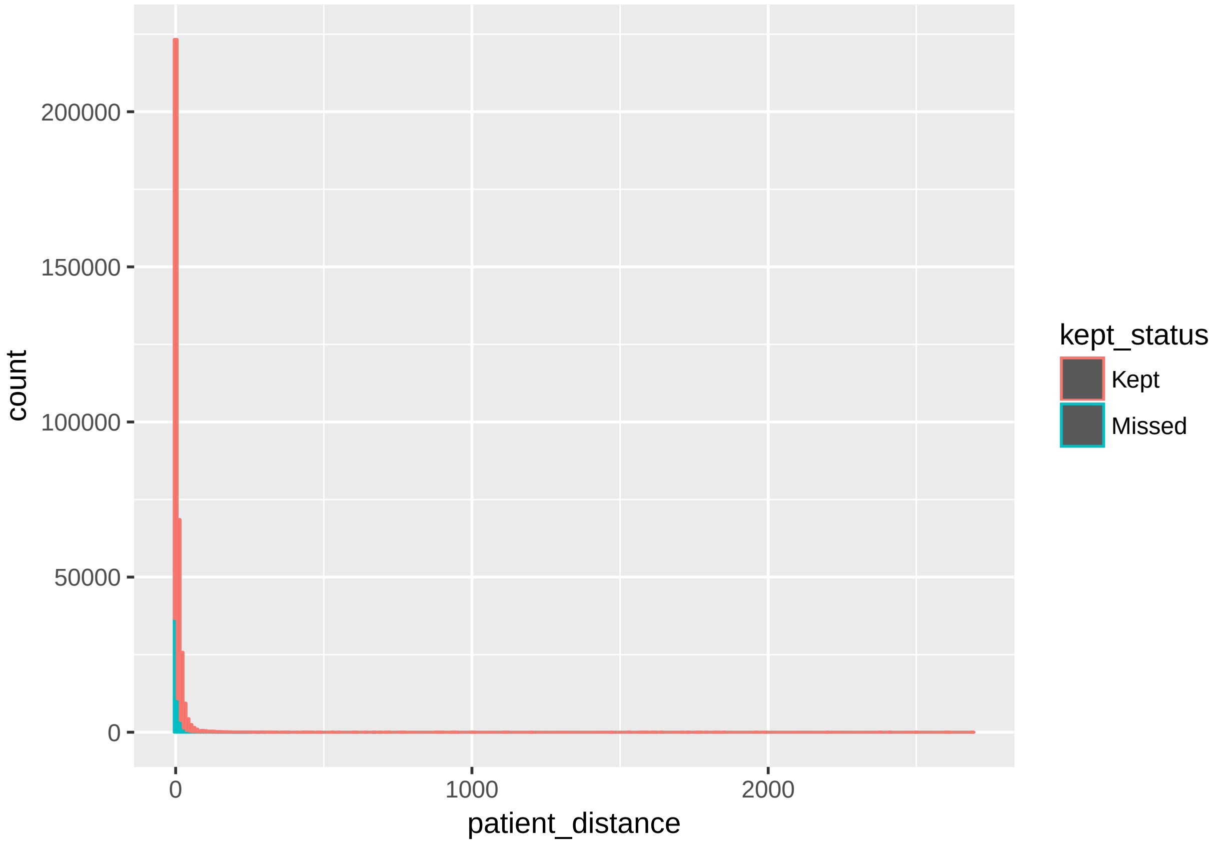
```
length_breaks <- c(-1, 45, 75, 1000)

length_labels <- c("Short", "Medium", "Long")

appointments <- appointments %>%
    mutate(
        length_group = cut(
            appt_length, breaks = length_breaks, labels = length_labels)
        )

ggplot(
    data = appointments,
    mapping = aes(x = length_group, fill = kept_status)
) +
    geom_bar(position = "fill")
```



patient__distance

```
ggplot(
    data = appointments,
    aes(x = patient_distance, group = kept_status, col = kept_status)
) +
    geom_histogram(binwidth = 10)
```

```
## Warning: Removed 972 rows containing non-finite values (stat_bin).
```



patient_distance is very right-skewed, therefore NA values will be replaced with median rather than mean.

```
appointments$patient_distance <- appointments$patient_distance %>%
    replace_na(median(appointments$patient_distance, na.rm = TRUE))
```

New Variables

The *percent_missed* variable is the percentage of prior appointments missed, calculated by dividing the prior missed appointments by the total number of prior appointments. For new patients, this calculation will result in an error because it will be attempting to divide by zero.

The variable *new* will specify whether a patient is new, represented by a 1, or existing, represented by 0. This is calculated by searching for appointments where *prior_missed* and *prior_kept* are both 0.

The variable *appt_lead_time* will calculate how far in advance an appointment was booked. This is calculated by taking the difference between *date_scheduled* and *appt_date*.

The variable *weekday* is simply the day of the week.

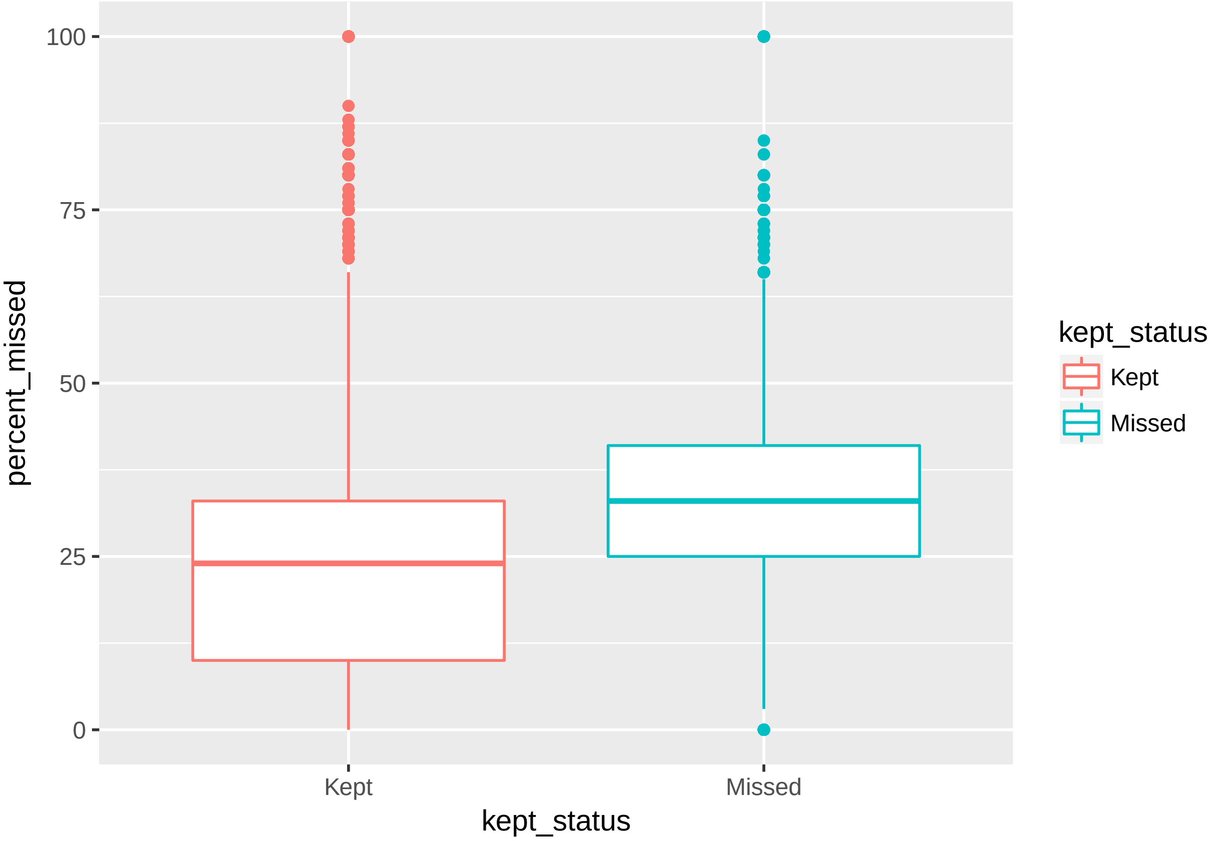
```
appointments <- appointments %>%
    mutate(percent_missed = prior_missed / (prior_missed + prior_kept)) %>%
    mutate(new = ifelse(prior_missed == 0 & prior_kept == 0, 1, 0)) %>%
    mutate(appt_lead_time = date(appt_datetime) - date(date_scheduled)) %>%
    mutate(weekday = strftime(appt_datetime, "%A"))

appointments$percent_missed <- as.integer(appointments$percent_missed * 100)
appointments$percent_missed <- appointments$percent_missed %>%
    tidyr::replace_na(mean(appointments$percent_missed, na.rm = TRUE))
```

Add county_code from zipcode data.

```
appointments <- dplyr::left_join(appointments, zipcodes, by = "office_zip")
```

percent_missed

```
ggplot(
    data = appointments,
    aes(x = kept_status, y = percent_missed, col = kept_status)
) +
    geom_boxplot()
```

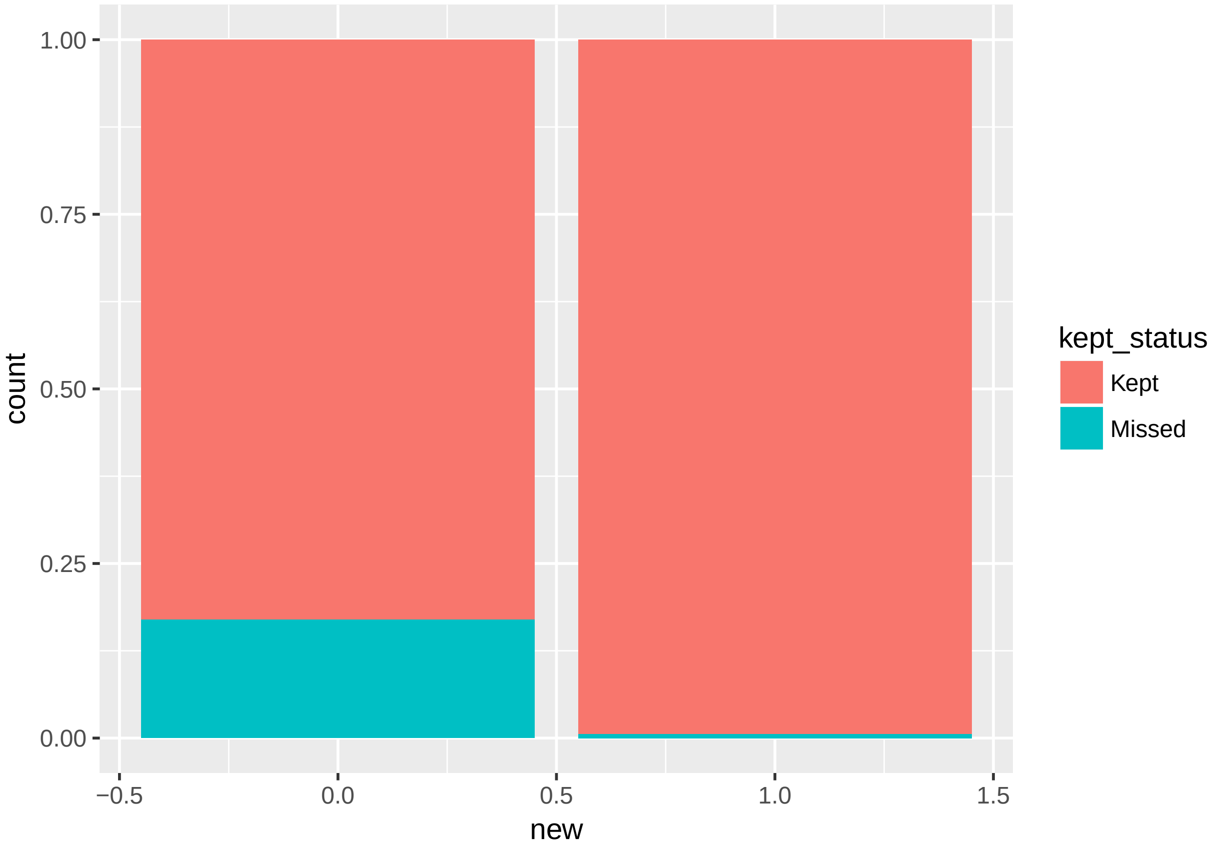


new

```
table(appointments$new)
```

```
##
##      0      1
## 320442  22340
```

```
ggplot(
    data = appointments,
    mapping = aes(x = new, fill = kept_status)
) +
    geom_bar(position = "fill")
```

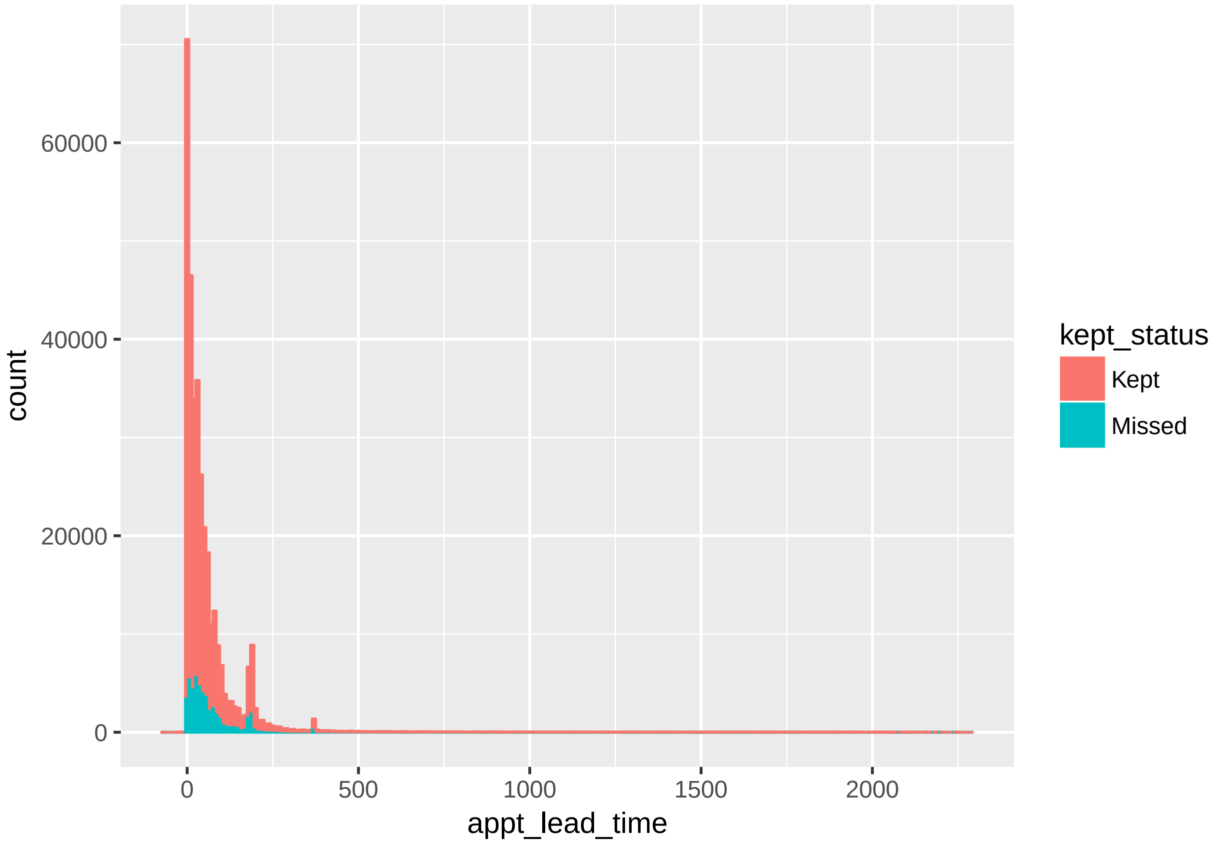


New patients have a very high percentage of kept appointments. 22k of 342k appointments are first-time, or about 6.4%

appt_lead_time

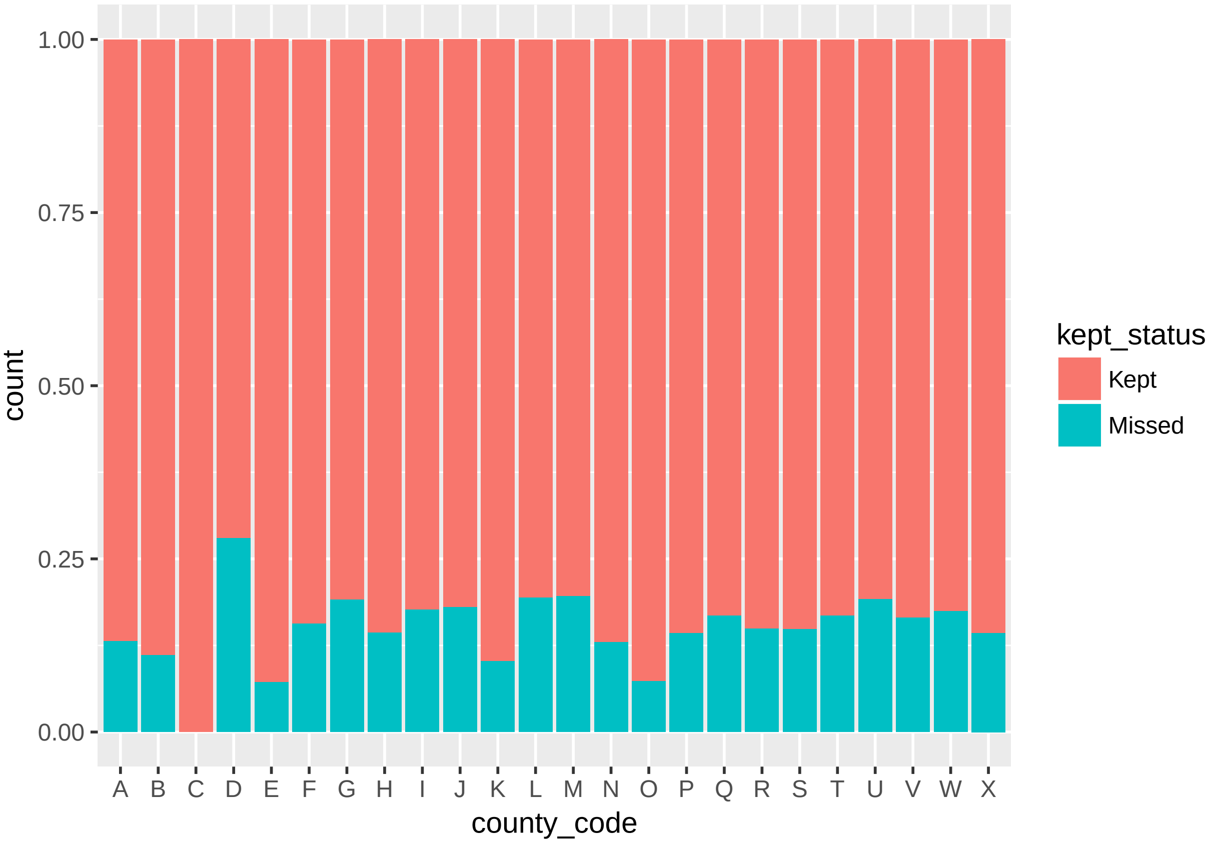
```
ggplot(
    appointments,
    aes(x = appt_lead_time, col = kept_status, fill = kept_status)
) +
    geom_histogram(binwidth = 10)
```

```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



county_code

```
ggplot(
    appointments,
    aes(x = county_code, fill = kept_status)
) +
    geom_bar(position = "fill")
```



```
#appointments_sample_025 <- appointments_3 %>%
#   sample_frac(size = 0.025, replace = FALSE)
#ggpairs(data = appointments_sample_05[,20:24], cardinality_threshold = 50)
```

Modeling

Create Modeling Data

```
model_data <- appointments

factor_columns <- c("kept_status", "patient_gender", "new", "billing_type",
"office_zip", "provider_specialty", "remind_call_result", "hour", "weekday",
"county_code", "length_group")
model_data[factor_columns] <- lapply(model_data[factor_columns], factor)
#Check for NAs
purrr::map_dbl(model_data, ~sum(is.na(.)))
```

```
##        kept_status          appt_date          appt_time
##                  0                  0                  0
##        appt_length     date_scheduled        patient_age
##                  0                  0                  0
##     patient_gender       billing_type       prior_missed
##                  0                  0                  0
##         prior_kept   patient_distance         office_zip
##                  0                  0                  0
## provider_specialty remind_call_result      appt_datetime
##                  0                  0                  0
##             missed            age_cat               hour
##                  0                  0                  0
##       length_group     percent_missed                new
##                  0                  0                  0
##     appt_lead_time            weekday        county_code
##                  0                  0                  0
```

```
model_data <- model_data %>%
    select(kept_status, patient_age, remind_call_result, provider_specialty,
           billing_type, hour, percent_missed, appt_length, patient_gender,
           patient_distance, new, appt_lead_time, weekday, county_code)
```

Divide model_data into train, validate, and test sets

```
train <- model_data[1:205660,]
validate <- model_data[205661:274200,]
test <- model_data[274201:nrow(model_data),]
table(train$kept_status)
```

```
##
##   Kept Missed
## 174600  31060
```

```
train2 <- train[168738:205660,]
table(train2$kept_status)
```

```
##
##   Kept Missed
##  31059   5864
```

```
train_kept <- train2[train2$kept_status == "Kept",]
train_missed <- train[train$kept_status == "Missed",]
# check out caret::downSample
train_balanced <- rbind(train_kept, train_missed)
table(train_balanced$kept_status)
```

```
##
##   Kept Missed
##  31059  31060
```

Logistic Regression Model

```
glm_train <- caret::train(kept_status ~ ., data = train_balanced, method = "glm")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

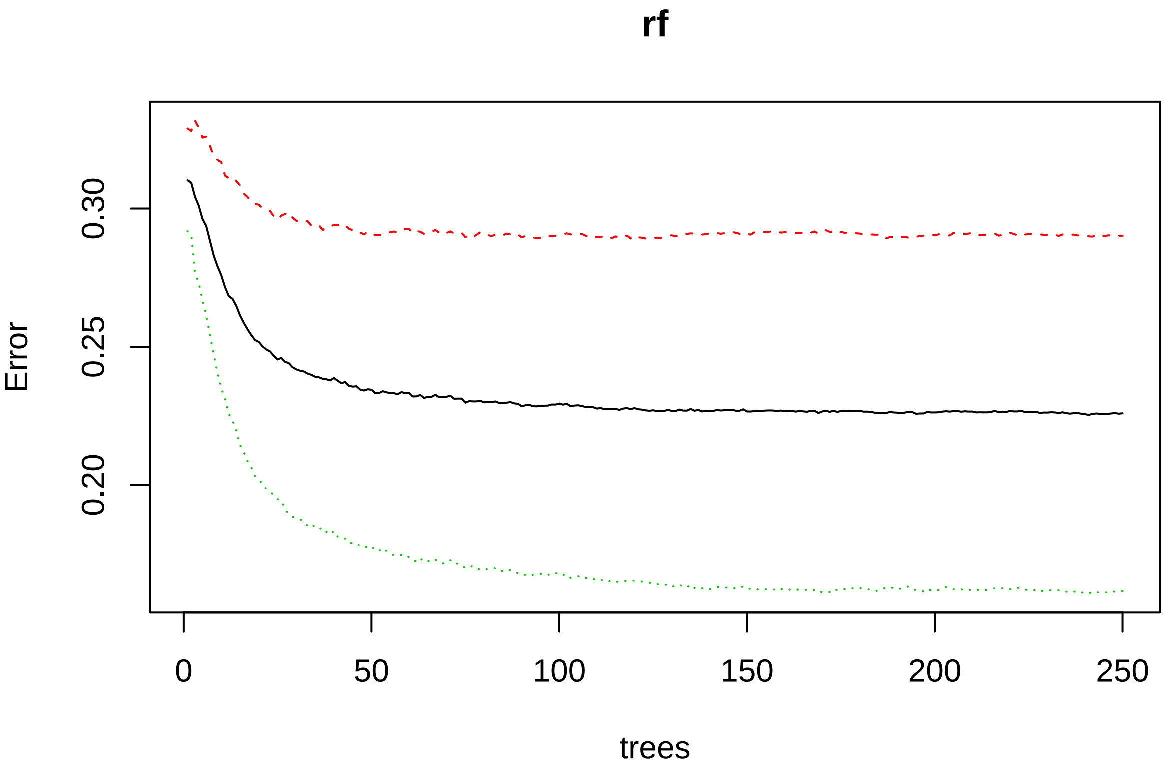
Random Forest Model

Using randomForest Package (To be removed in final report)

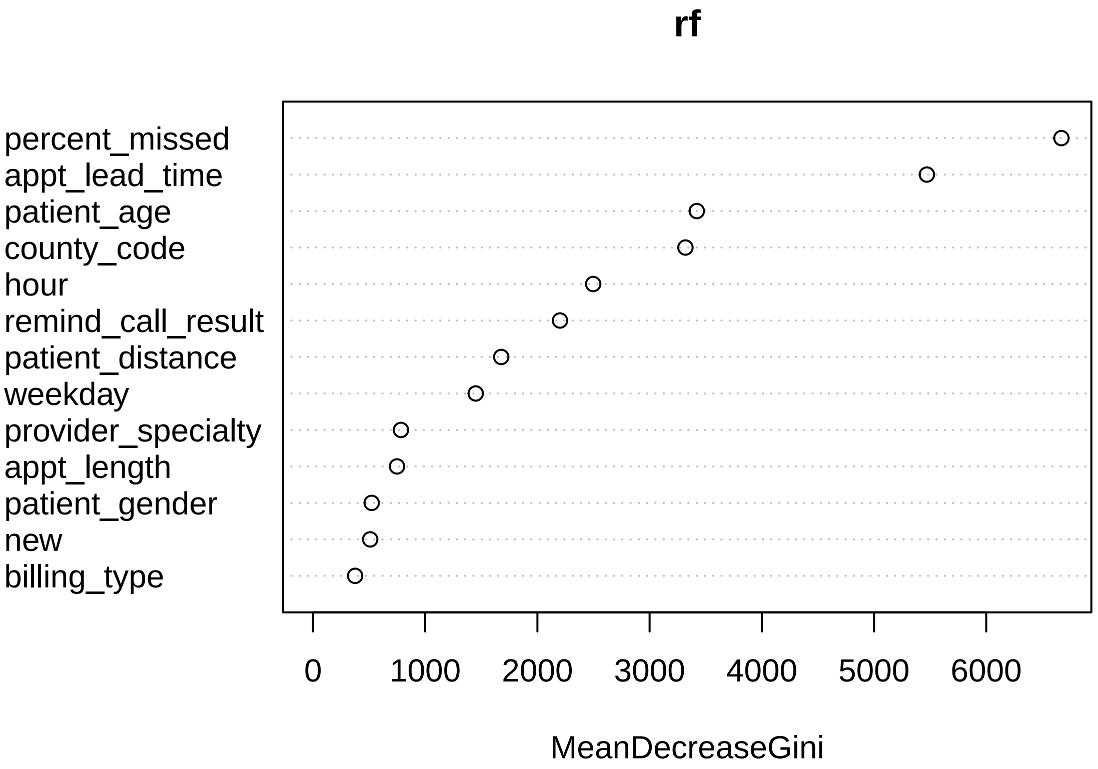
```
rf <- randomForest(kept_status ~ ., data = train_balanced, ntree = 250)

print(rf)
```

```
##
## Call:
##  randomForest(formula = kept_status ~ ., data = train_balanced,      ntree = 250)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 22.59%
## Confusion matrix:
##         Kept Missed class.error
## Kept   22047   9012   0.2901574
## Missed  5022  26038   0.1616871
```

```
plot(rf)
```



```
varImpPlot(rf)
```



Using caret Package

```
# Try adding classProbs = TRUE
control <- caret::trainControl(method = "cv", number = 2)
```

```
seed <- 7
metric <- "Accuracy"
set.seed(seed)
mtry <- 3
tunegrid <- expand.grid(.mtry = mtry)
```

```
rftrain <- caret::train(kept_status ~ ., data = train_balanced, method = "rf", metric = metric, tuneGri
```

Model Comparison

```
caret::confusionMatrix(glm_train)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction Kept Missed
##     Kept   35.3   11.4
##     Missed 14.6   38.7
##
##  Accuracy (average) : 0.7403
```

```
pred_glm <- predict(glm_train, validate)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
conf_mat_glm <- caret::confusionMatrix(pred_glm, validate$kept_status, positive = "Missed")
conf_mat_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Kept Missed
##     Kept   41896   3127
##     Missed 15189   8328
##
##                Accuracy : 0.7328
##                  95% CI : (0.7294, 0.7361)
##     No Information Rate : 0.8329
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3244
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7270
##             Specificity : 0.7339
##          Pos Pred Value : 0.3541
##          Neg Pred Value : 0.9305
##              Prevalence : 0.1671
##          Detection Rate : 0.1215
##    Detection Prevalence : 0.3431
##       Balanced Accuracy : 0.7305
```

```
##
##        'Positive' Class : Missed
##
```

```
conf_mat_glm$byClass["F1"]
```

```
##        F1
## 0.4762667
```

```
pred_rf <-predict(rftrain, validate)

caret::confusionMatrix(rftrain)
```

```
## Cross-Validated (2 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction Kept Missed
##     Kept   33.8    8.5
##     Missed 16.2   41.5
##
##  Accuracy (average) : 0.7538
```

```
conf_mat_rf <- caret::confusionMatrix(pred_rf, validate$kept_status, positive = "Missed")
conf_mat_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Kept Missed
##     Kept   38990   2348
##     Missed 18095   9107
##
##                Accuracy : 0.7017
##                  95% CI : (0.6983, 0.7052)
##     No Information Rate : 0.8329
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3085
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7950
##             Specificity : 0.6830
##          Pos Pred Value : 0.3348
##          Neg Pred Value : 0.9432
##              Prevalence : 0.1671
##          Detection Rate : 0.1329
##    Detection Prevalence : 0.3969
##       Balanced Accuracy : 0.7390
##
##        'Positive' Class : Missed
##
```

```
conf_mat_rf$byClass["F1"]
```

```
##        F1
## 0.4711695
```

```
###glm currently performing slightly better than rf on validation data based on F1 score
```