# Capstone Project - Predicting Patient No-Shows Using Appointment Data

*Derek Samsom*

Missed medical appointments are a major problem in the medical industry, resulting in lost revenue. Medical providers can over-book appointments to try to minimize the lost revenue, but without any way to predict the probability of an appointment being missed, there will still be times where more or fewer patients show up at a given time than expected. The result will be that lost revenue will be reduced but not eliminated, as there will still be times that more appointments are missed than expected. There will aslo be times more apppointments show up than expected, which can overwhelm staff and resources and affect the level of patient care.

This project is a classification problem that will explore the prediction of whether a medical appointment will be missed, and its probability of being kept or missed. The prediction error will result in times where there are too many or too few patients at a given time. The main goal in the prediction will be to minimize the error, as this will reduce instances of having more or fewer patients that desired.

There are countless reasons and circumstances that can lead someone to miss an appointment, such as a last minute work meeting or a family emergency, that aren't directly captured in the data and are impossible to know in advance. Missed appointments can only be predicted based based on indirect factors that are known, such as past history as demographics. Because of this, there will be a level of error that cannot be eliminated, however, any reduction in error compared to having no predictive model at all is still beneficial.

Medical providers can used the missed appointment predistions by incorporating them into their booking methods and systems. The methods used in booking will have to consider the implications of the inherent prediction errors and balance the risk the errors represent: too many patients leading to staff/resourse shortage, and too few patients leading to lost revenue. The methods of implementing the use of missed appointentment predictions into an appointment booking system are client-specific ant not included in the scope of this project, which is limited to minimizing the error while predicting the probability that an appointment will be missed or kept.

I will start off by loading the required packages and the data.

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

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, pre-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, `zipcodes`, has information about the county the offices are located in. This will be used to see if the location 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
##
##
##
##
```

```r
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 2880
##  $ 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_descriptions <- 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_descriptions))
```

```
## # A tibble: 14 x 3
##    V1    variable           variable_descriptions
##    <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
```

The `appt_date` and `appt_time` variables can be combined into one variable, `appt_datetime`.

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

appointments$date_scheduled <- lubridate::as_date(
    appointments$date_scheduled, format = "%m/%d/%y", tz = "UTC")
```

## Data Exploration

First I want to calculate the percent of missed appointments overall by creating a logical variable `missed`, where 1 represents a missed appointment and 0 represents a kept appointment. This will determine the degree of class imbalance.

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

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

15.93% of the total appointments are missed. This is an imbalanced classification, which will have implications in the modeling. For example, the model could predict all of the appointments will be kept and be correct 84.07% of the time. This results in a high accuracy without providing any useful prediction of which appointments will be missed.

Next I want to check the data to see if there are any missing values that could indicate reduced data integrity or adversely affect the modelling.

```
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
```

One variable, `patient_distance` has 974 missing value. This is fairly minor and will be evaluated later on when exploring the variable further.

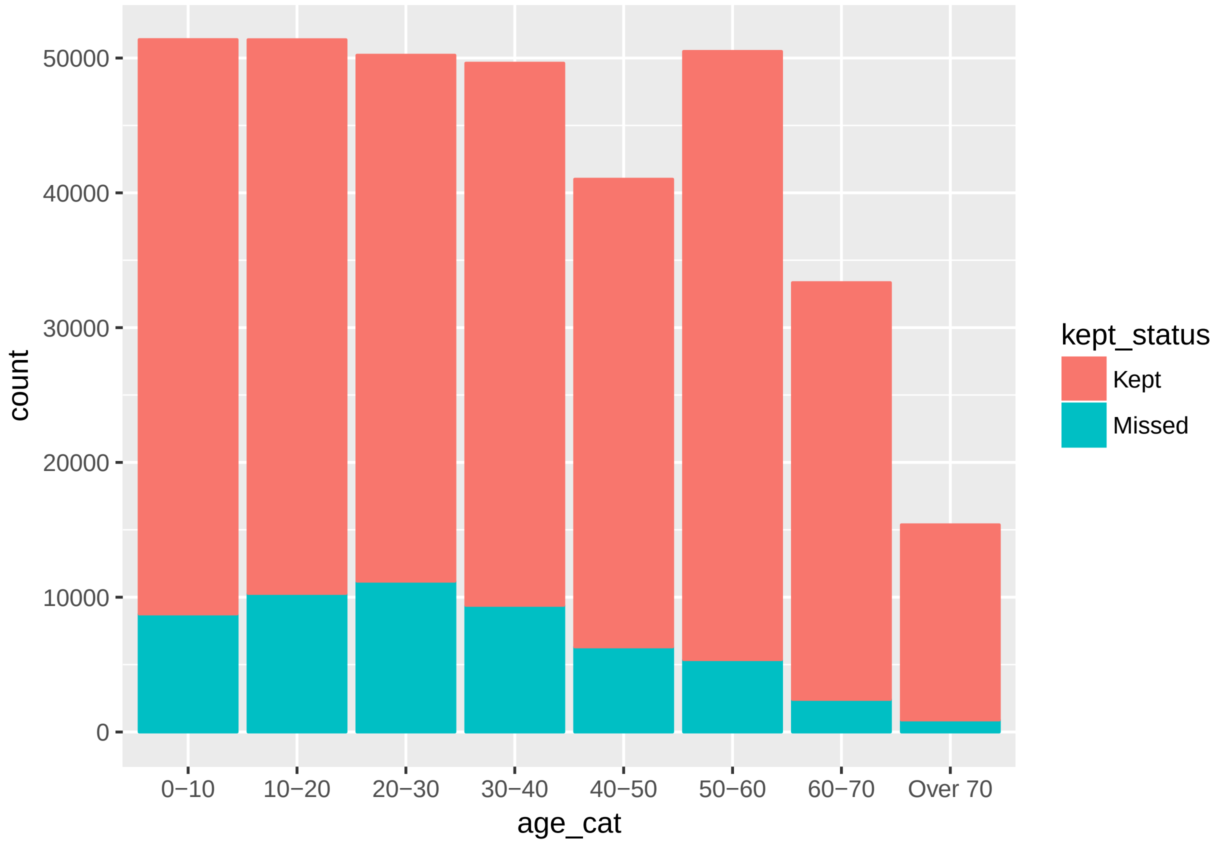
### patient_age

I expected missed appointments to have to vary across age ranges. Perhaps older patients have fewer commitments with kids or work, and make their appointments more regularly, or perhaps younger adults might skip more appointments because they aren't as critical? I will break the data into age groups to make the plot simpler to evaluate.

There are a small number of observations where the age is higher than plausible. Therefore, the observations greater than age 110 will be removed from the data.

```
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))
```

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




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

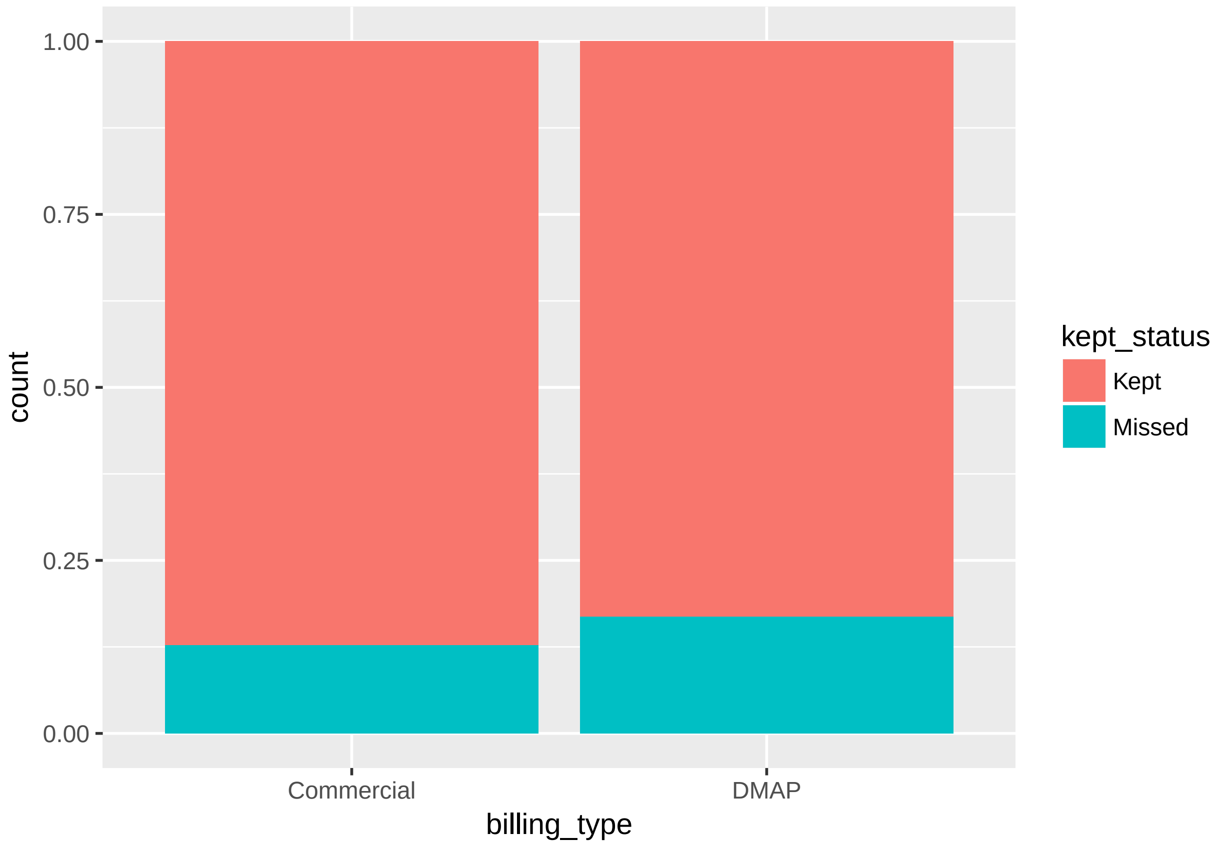
**billing_type**

```r
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.

```r
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 than commercial.

### appt_datetime

For the variable `appt_datetime`, I will create an `hour` variable to see the variation in missed appointments by hour of day.

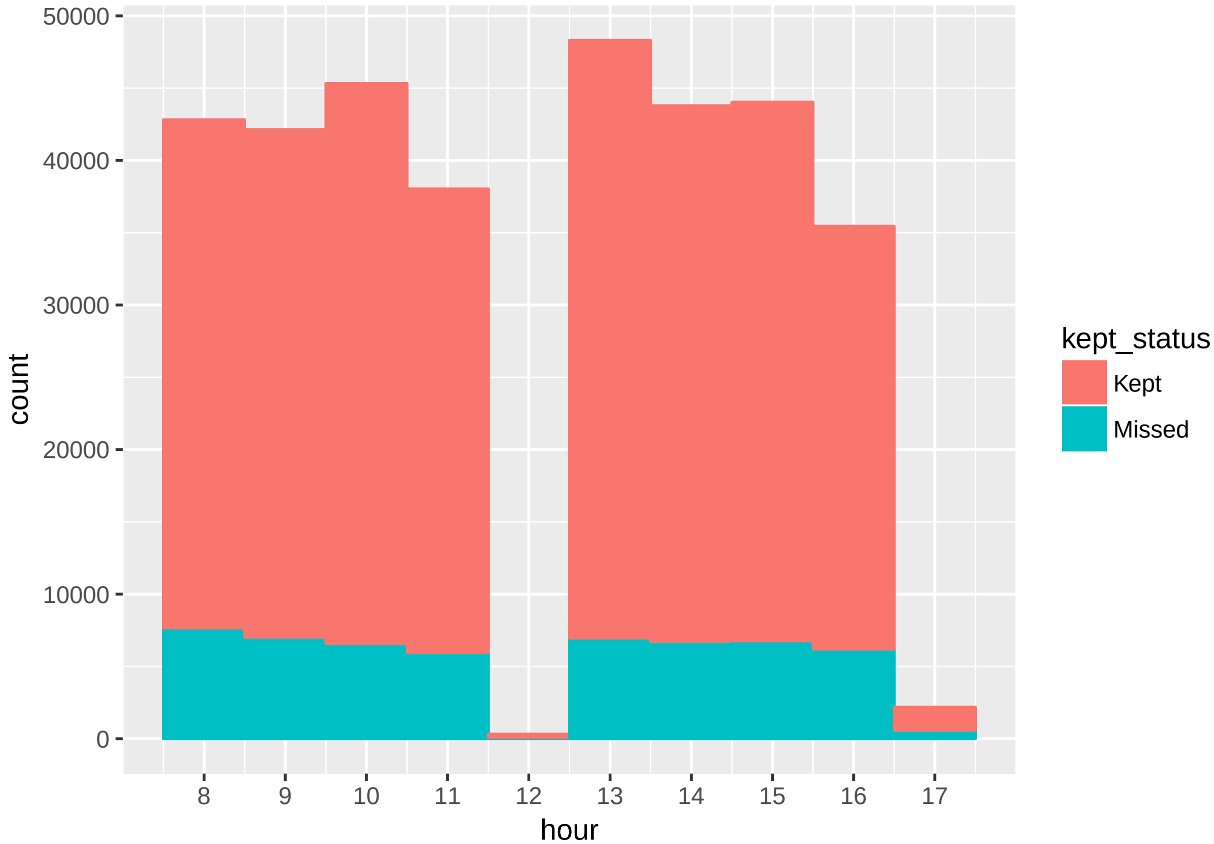
```
appointments <- appointments %>%
    mutate(hour = lubridate::hour(appointments$appt_datetime))

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
```

Most appointments are scheduled between 8:00 AM and 5:00 PM, with an hour gap starting at 12:00.

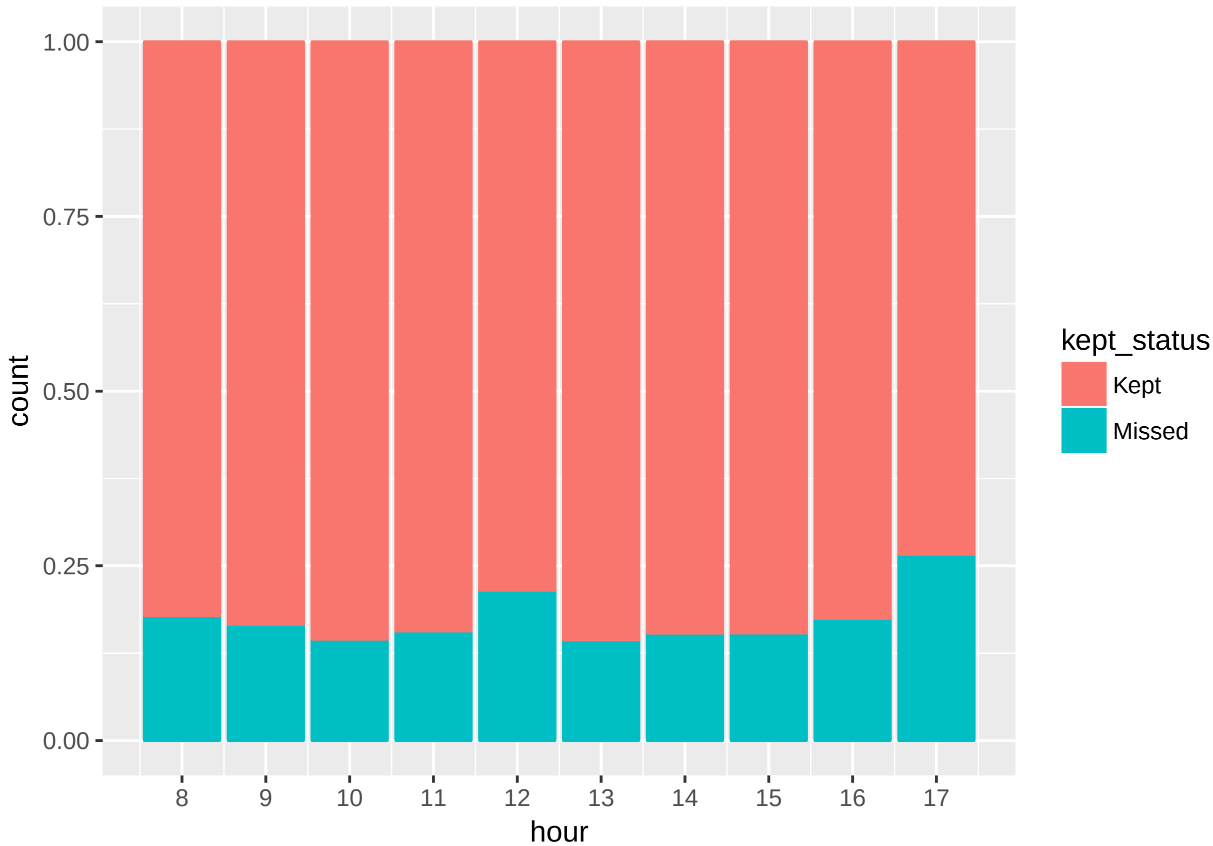
```
appointments_hour <- appointments %>%
    select(kept_status, hour) %>%
    filter(hour >= 8 & hour <= 17)

ggplot(
    appointments_hour,
    aes(x = hour, col = kept_status, fill = kept_status)
```

```
) +
    geom_histogram(binwidth = 1) +
    scale_x_continuous(breaks = seq(8, 17, 1))
```




There is a decline in the total number of missed appointments as both the morning and afternoon period progress, however, there are fewer appointments towars the end of the two periods.

```
ggplot(
    appointments_hour,
    aes(x = hour, col = kept_status, fill = kept_status)
) +
    geom_bar(position = "fill") +
    scale_x_continuous(breaks = seq(8, 17, 1))
```




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

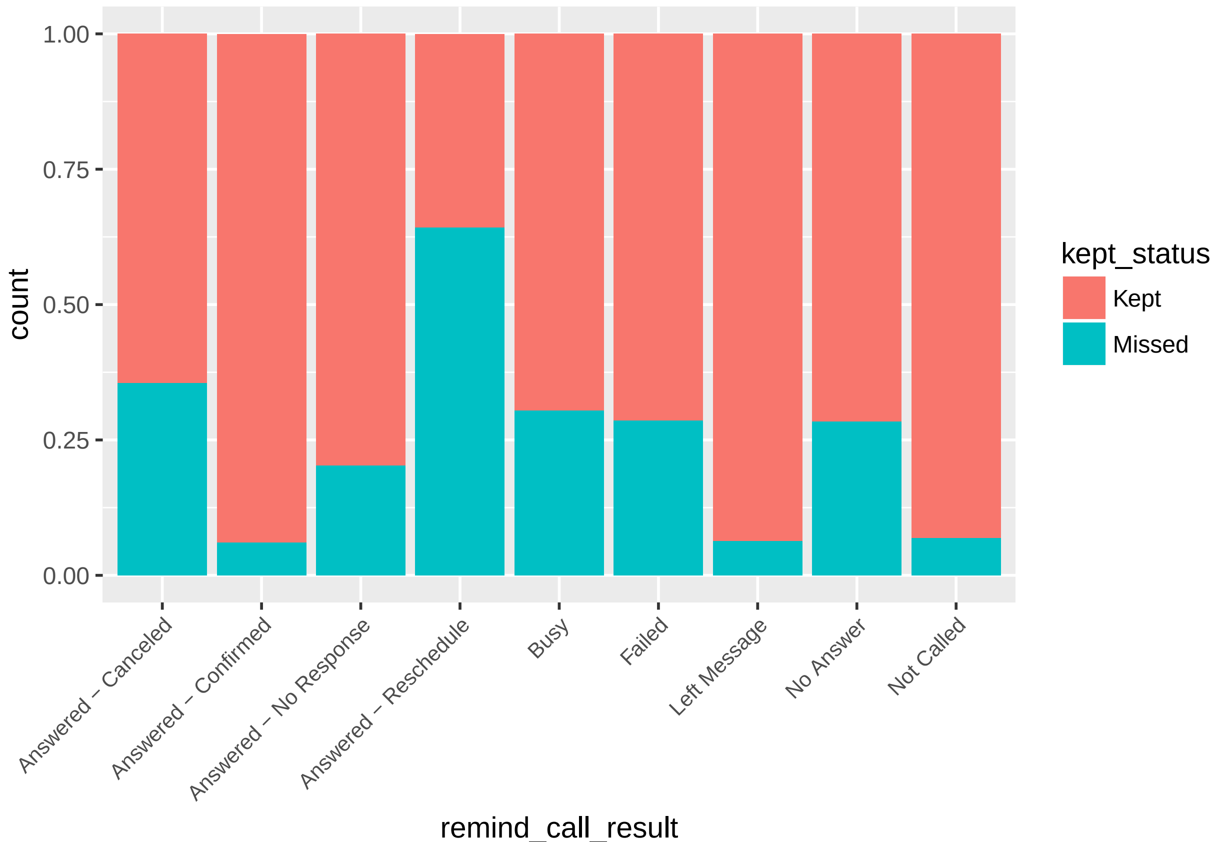
**remind_call_result**

```
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"

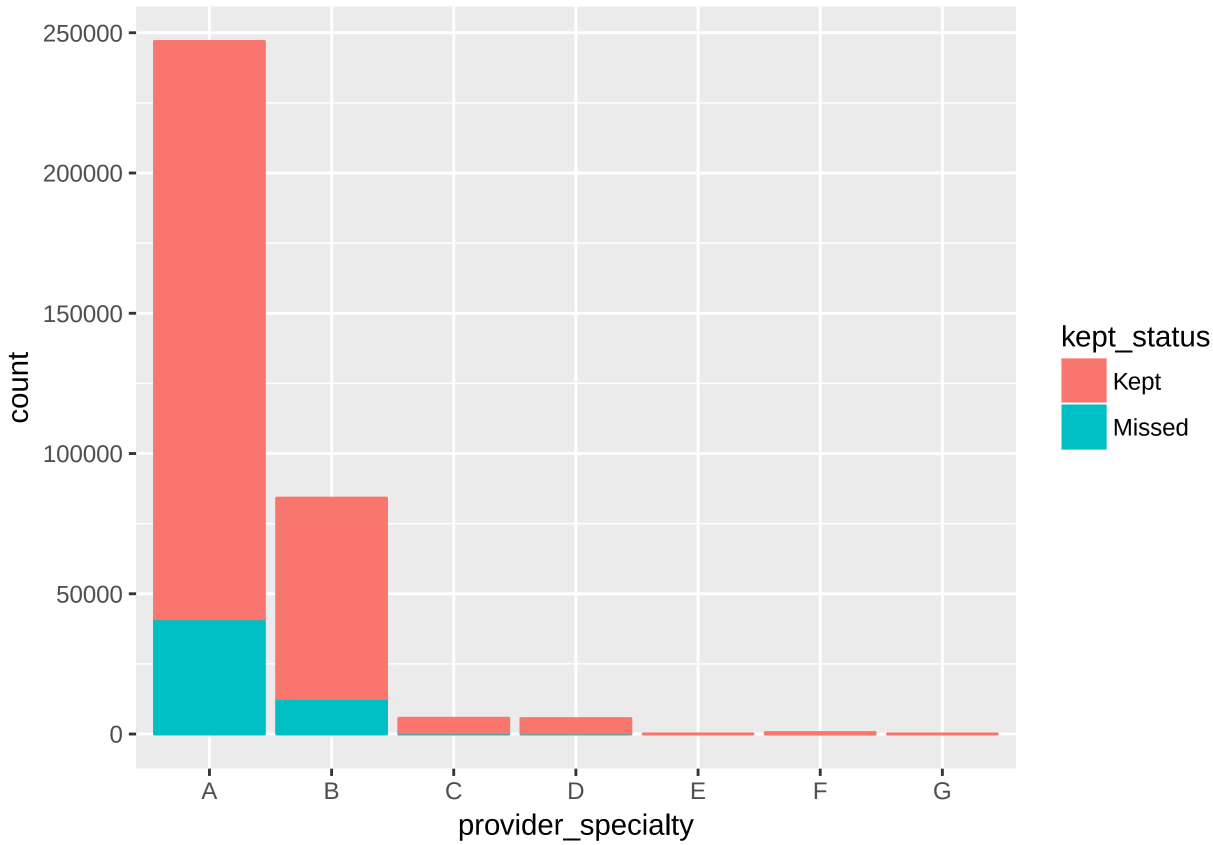
```
ggplot(
    appointments,
    aes(x = remind_call_result, fill = kept_status)
) +
    geom_bar(position = "fill") +
    theme(axis.text.x = element_text(size = 8, angle = 45,hjust = 1, vjust = 1))
```



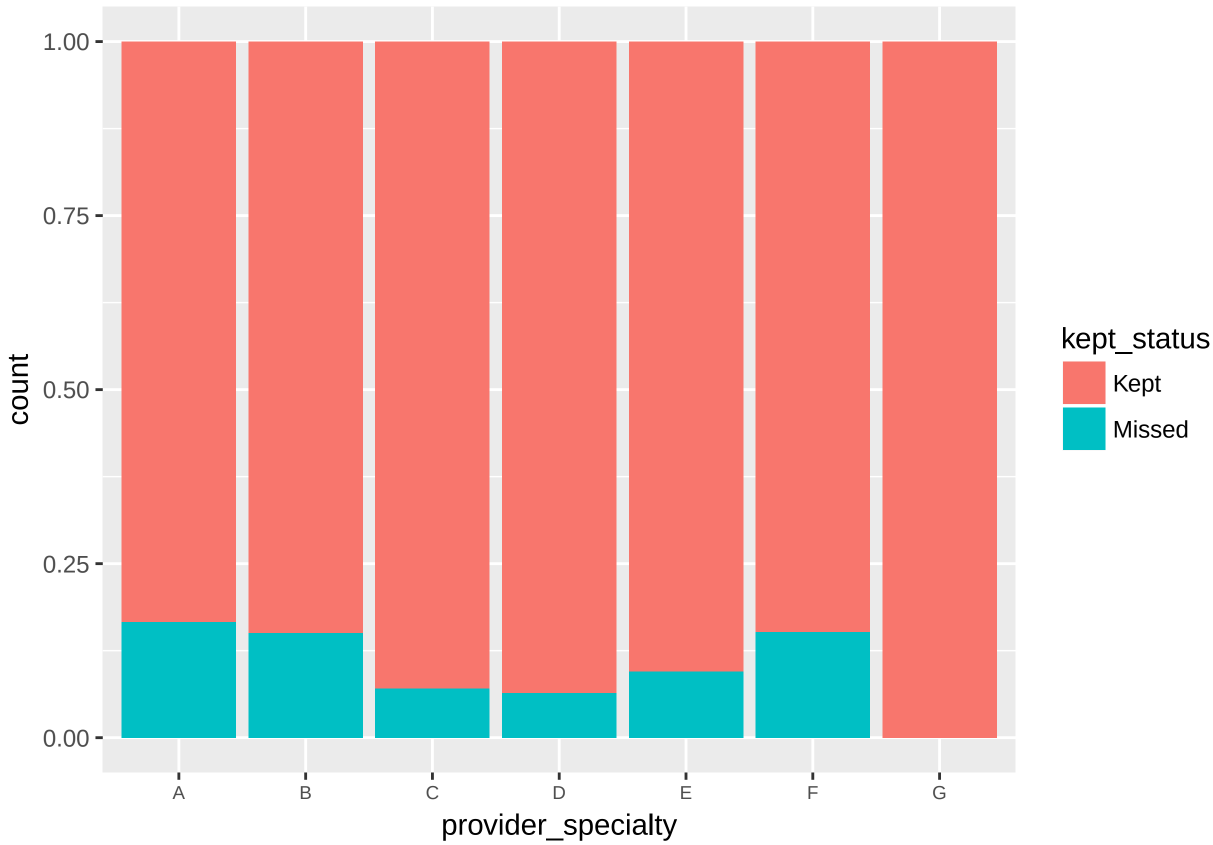

~65% of appointments with "Answered - Cancelled" and ~35% with "Answered-Reschedule" still kept their appointments, however, very few observations in these categories.

**provider_specialty (Should I even include this in the report? not as interesting since I have the specialties encoded)**

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



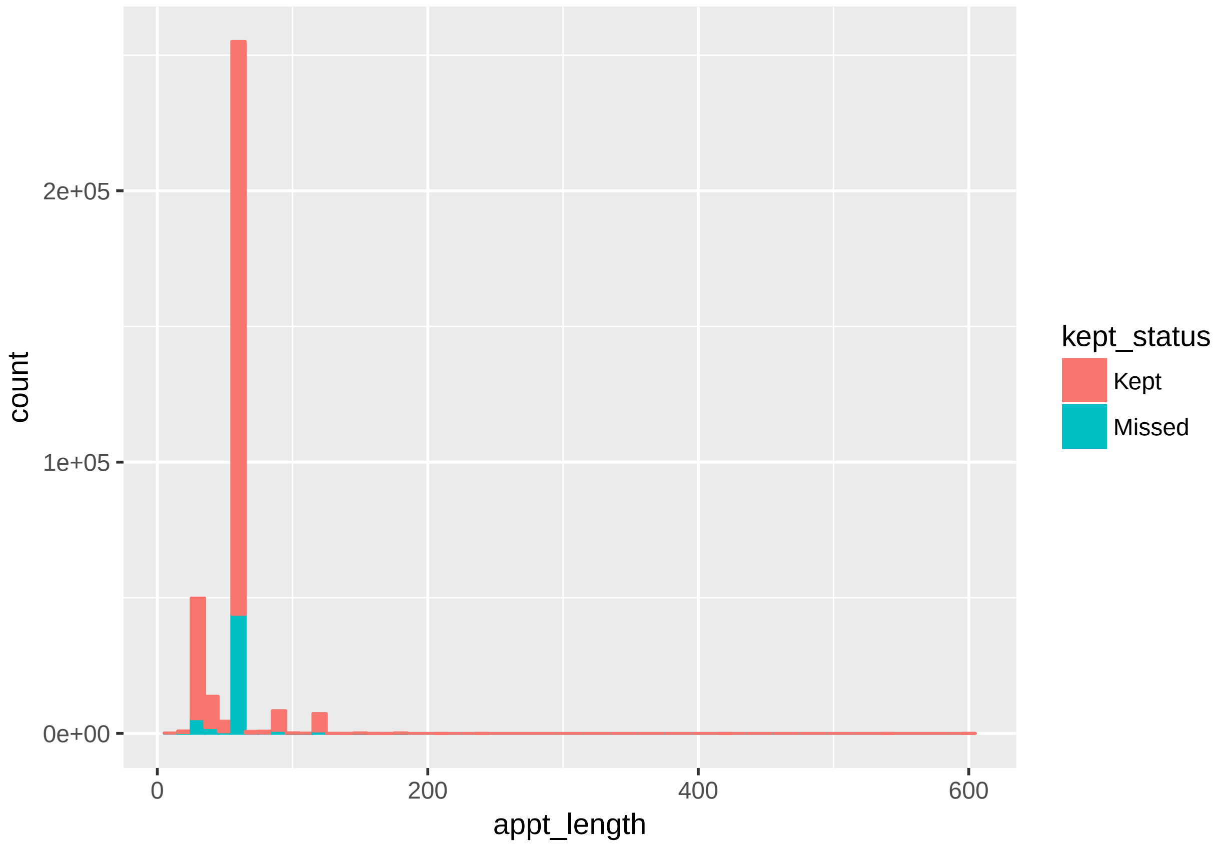

```r
ggplot(
    appointments,
    aes(x = provider_specialty, fill = kept_status)
) +
    geom_bar(position = "fill") +
    theme(axis.text.x = element_text(size = 7))
```




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

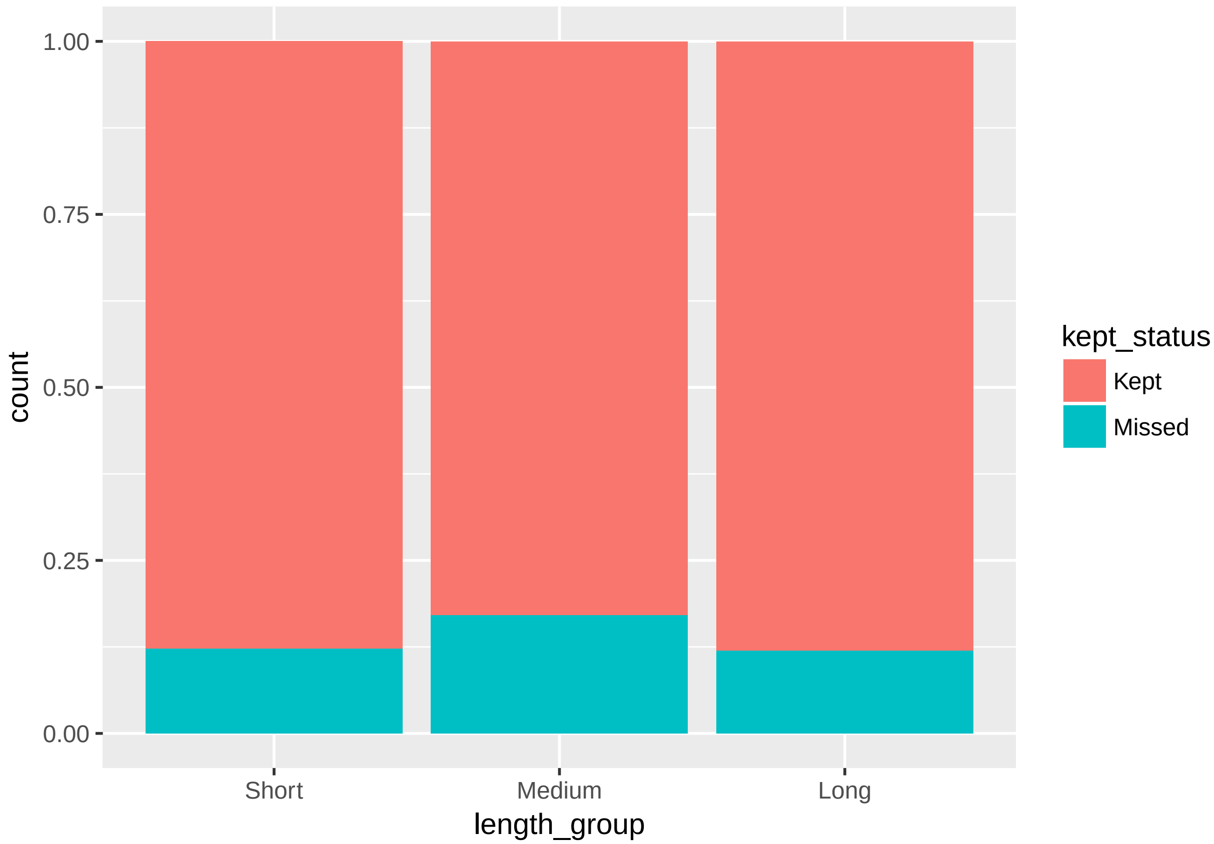
**appt__length**

```
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.

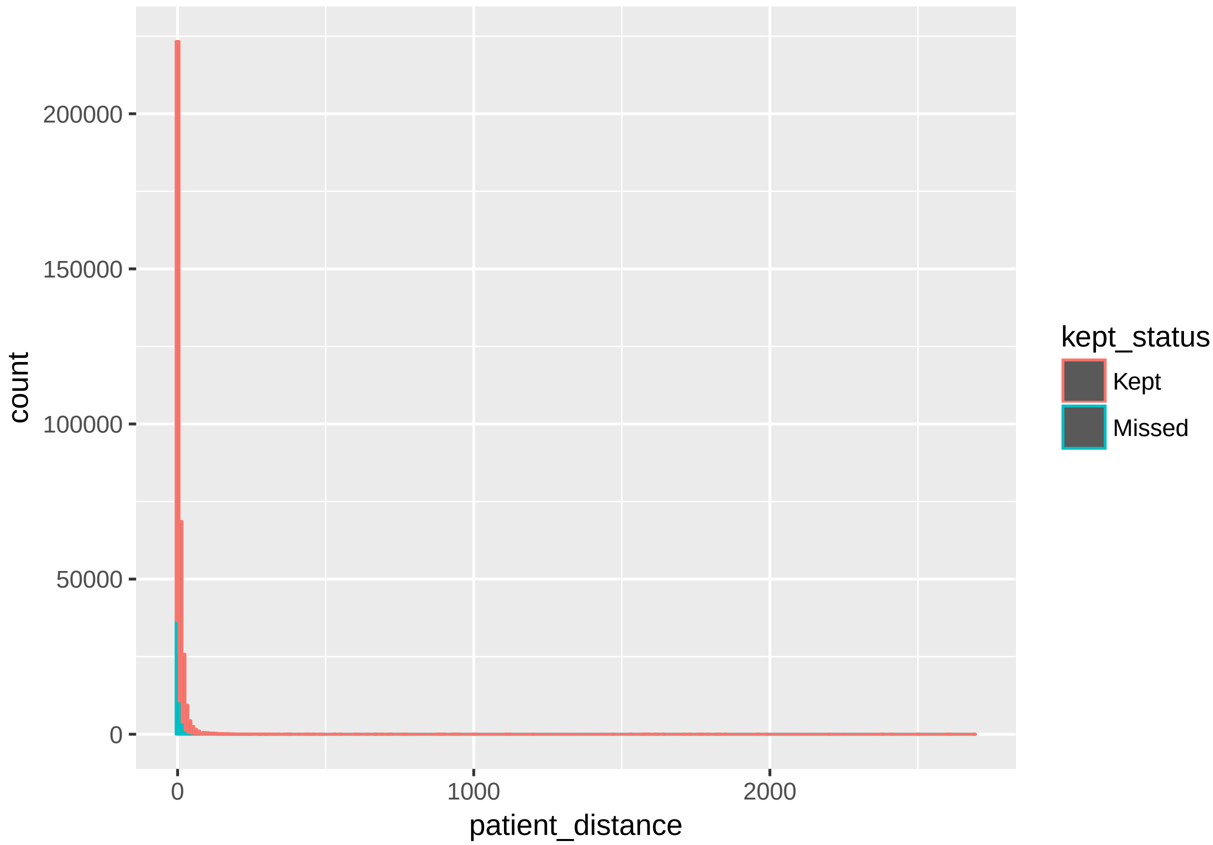
```
length_breaks <- c(-1, 45, 75, 1000)

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

In addition to the original variables, there are several additional variables that can be calculated based on the originals.

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 `is_new_patient` will specify whether a patient is new, represented by a 1, or existing, represented by 0. My hypothesis is that new patients are more likely to keep their appointments, since I think it is human nature to try to give a good first impression. 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`. If people are more likely to forget appointments booked farther in advance, or they are more likely to be for less urgent preventative care than last minute appointnemtns, this will pick that up.

The variable `appt_weekday` is the day of the week the appointments occurs, and `weekday_scheduled` is the date the appointment was booked.

```
appointments <- appointments %>%
    mutate(percent_missed = prior_missed / (prior_missed + prior_kept)) %>%
```

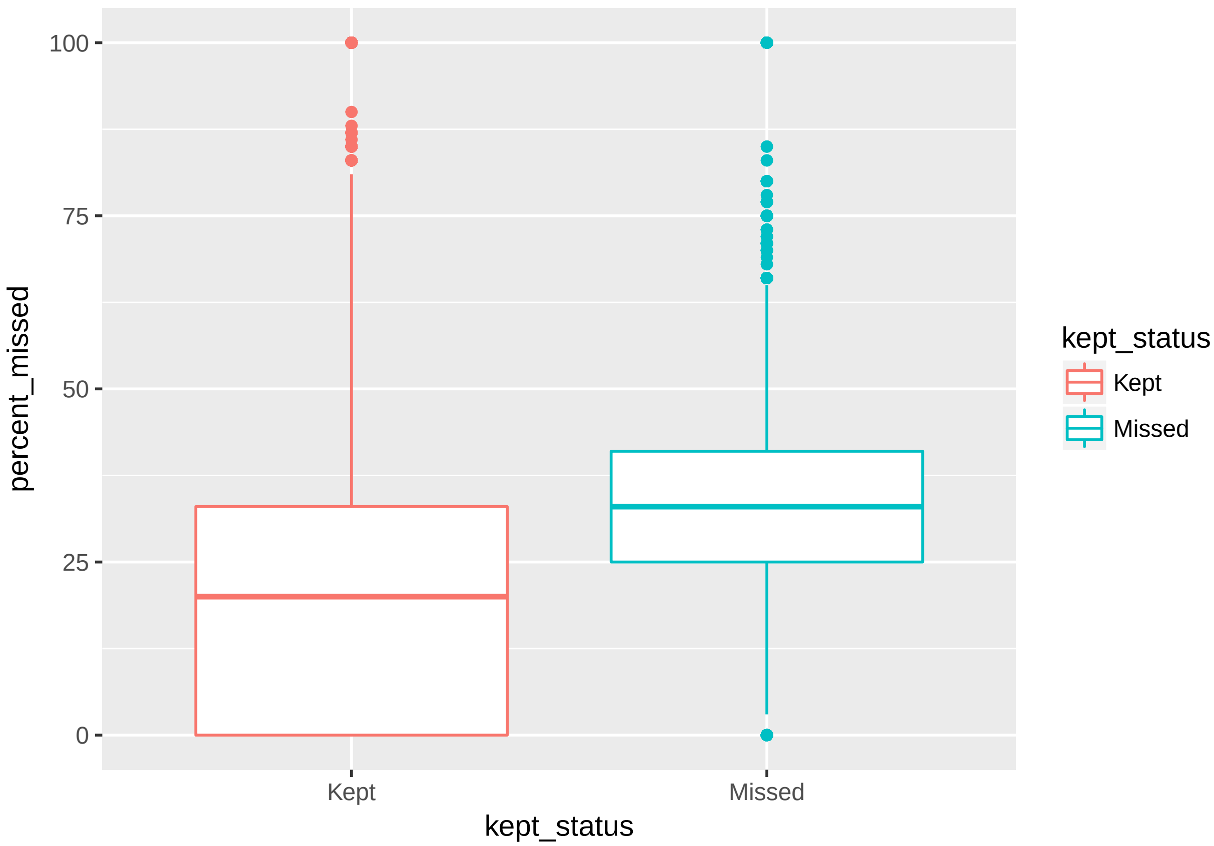
```
    mutate(
        is_new_patient = ifelse(prior_missed == 0 & prior_kept == 0, 1, 0)) %>%
    mutate(appt_lead_time = date(appt_datetime) - date(date_scheduled)) %>%
    mutate(appt_weekday = strftime(appt_datetime, "%A")) %>%
    mutate(weekday_scheduled = strftime(date_scheduled, "%A"))

appointments$percent_missed <- as.integer(appointments$percent_missed * 100)
appointments$percent_missed <- appointments$percent_missed %>%
    tidyr::replace_na(0)
```

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()
```

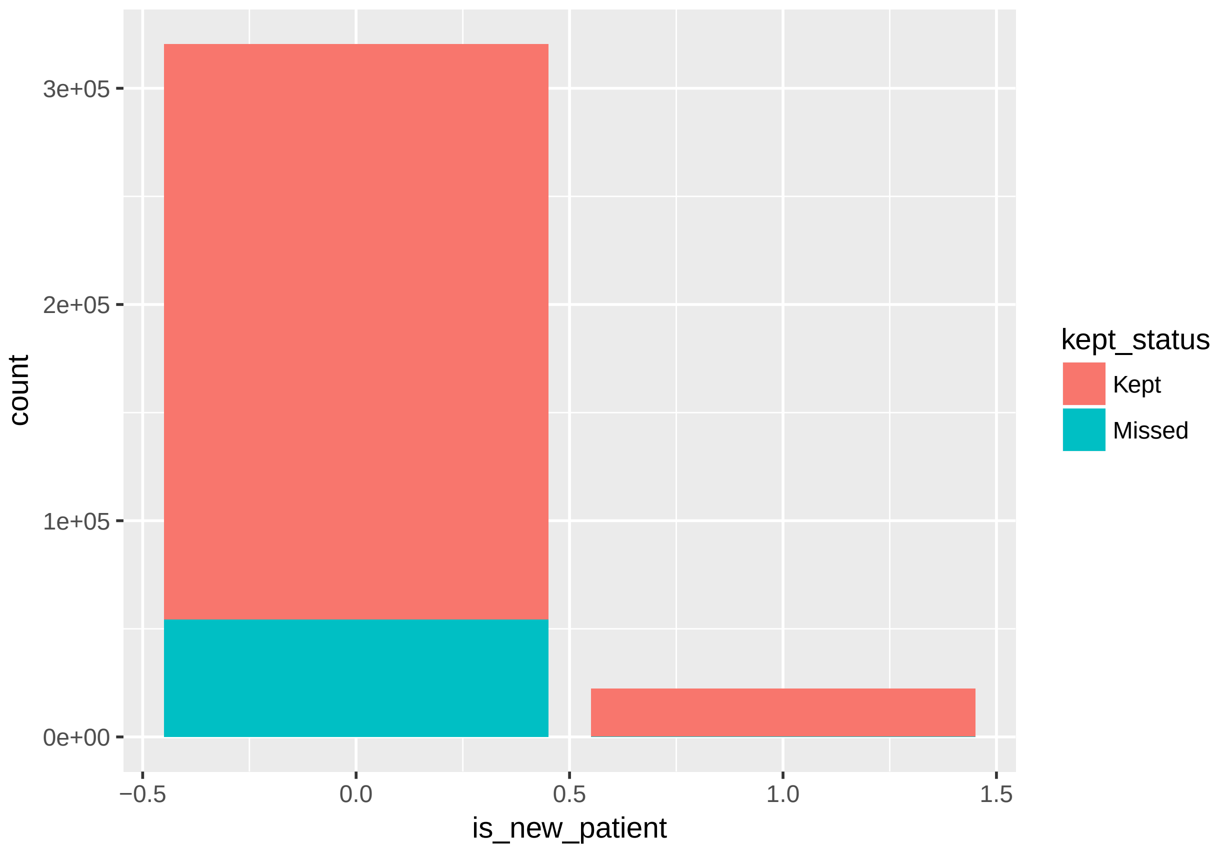


**is_new_patient**

```r
table(appointments$is_new_patient)
```

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

```r
ggplot(
    appointments,
    aes(x = is_new_patient, fill = kept_status)
) +
    geom_bar()
```

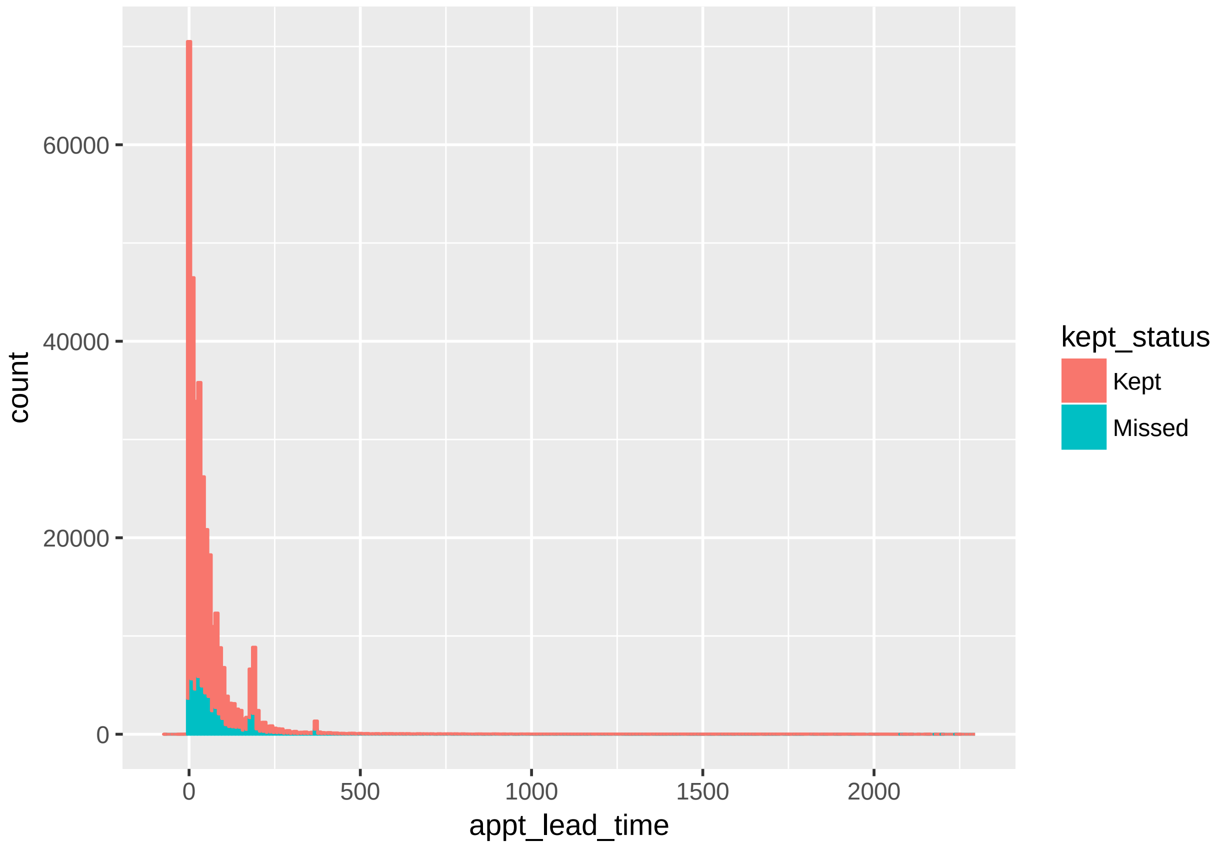



New patients have a very high percentage of kept appointments. 22,000 of 342,000 appointments are first-time, or about 6.4%

**appt_lead_time**

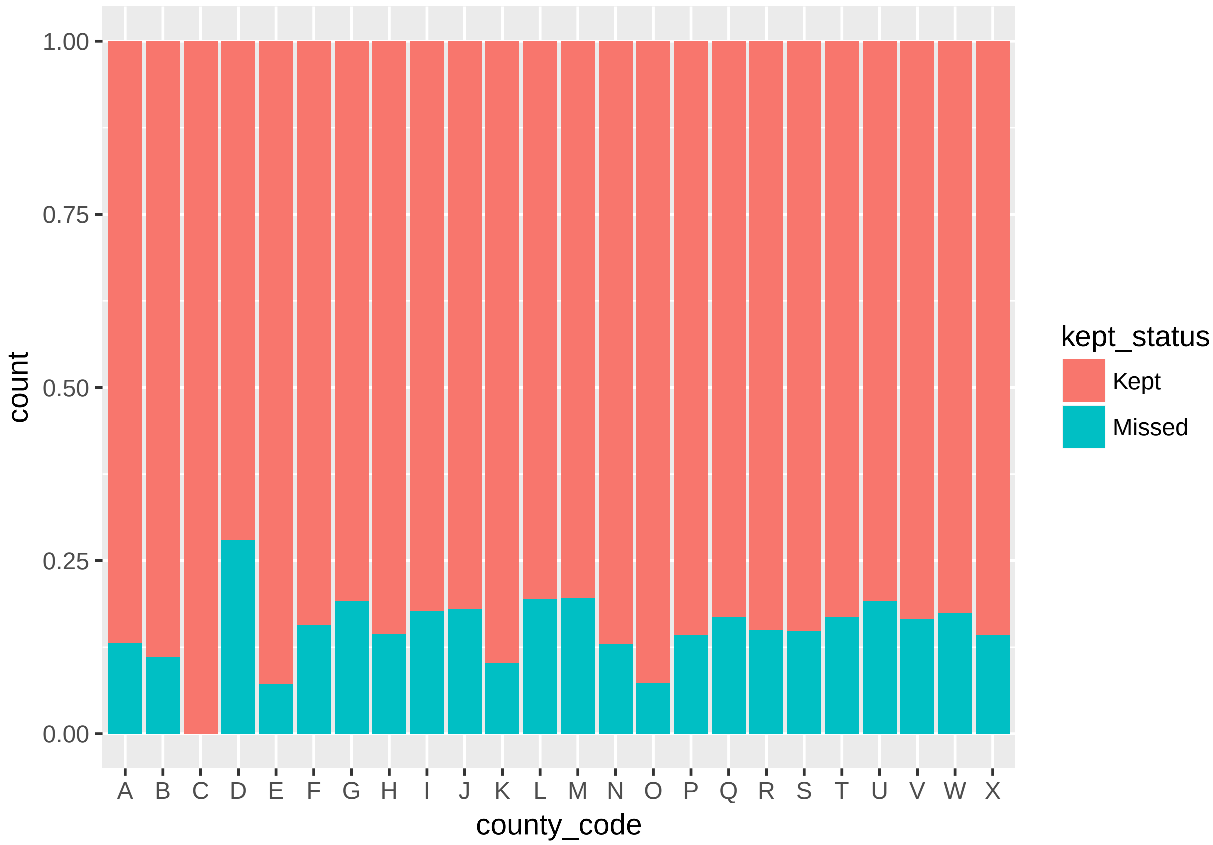
```r
ggplot(
    appointments,
    aes(x = appt_lead_time, col = kept_status, fill = kept_status)
) +
    geom_histogram(binwidth = 10)
```

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




**county_code**

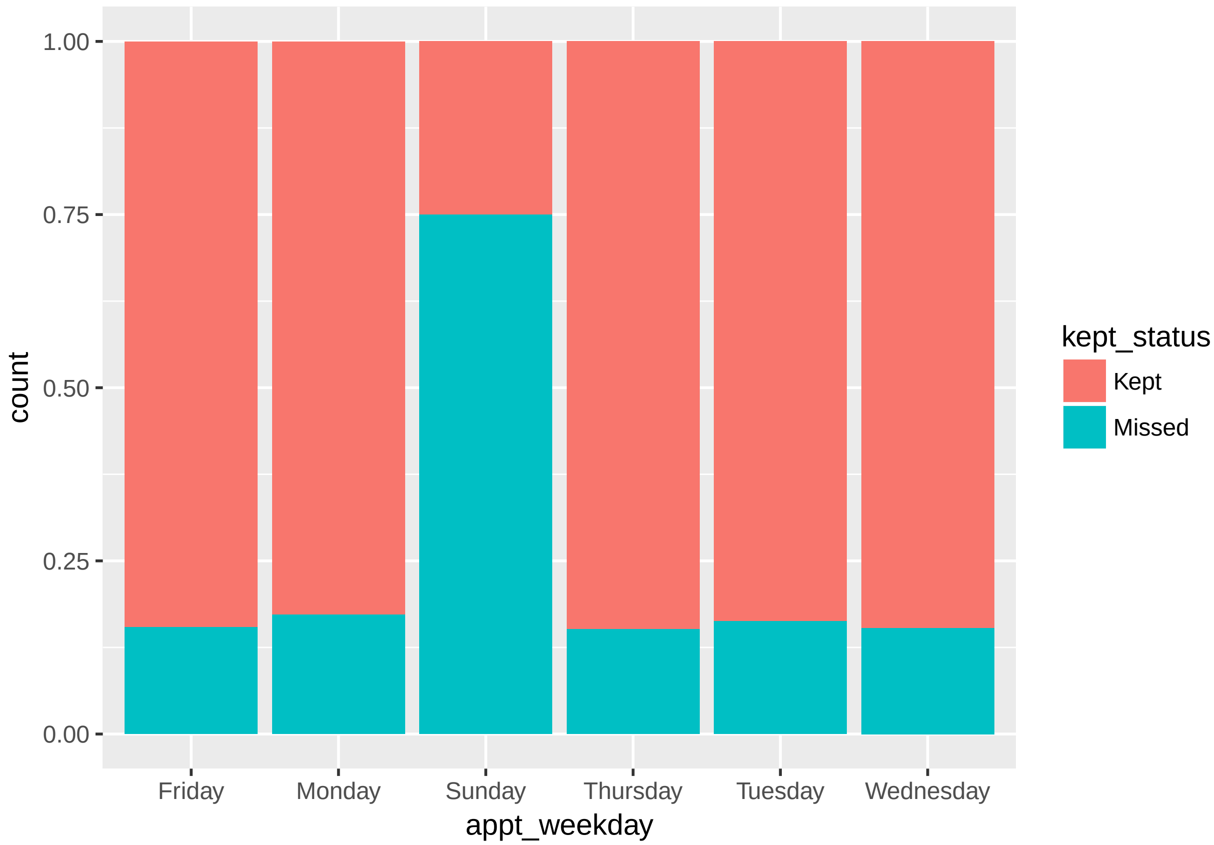
```
ggplot(
    appointments,
    aes(x = county_code, fill = kept_status)
) +
    geom_bar(position = "fill")
```




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

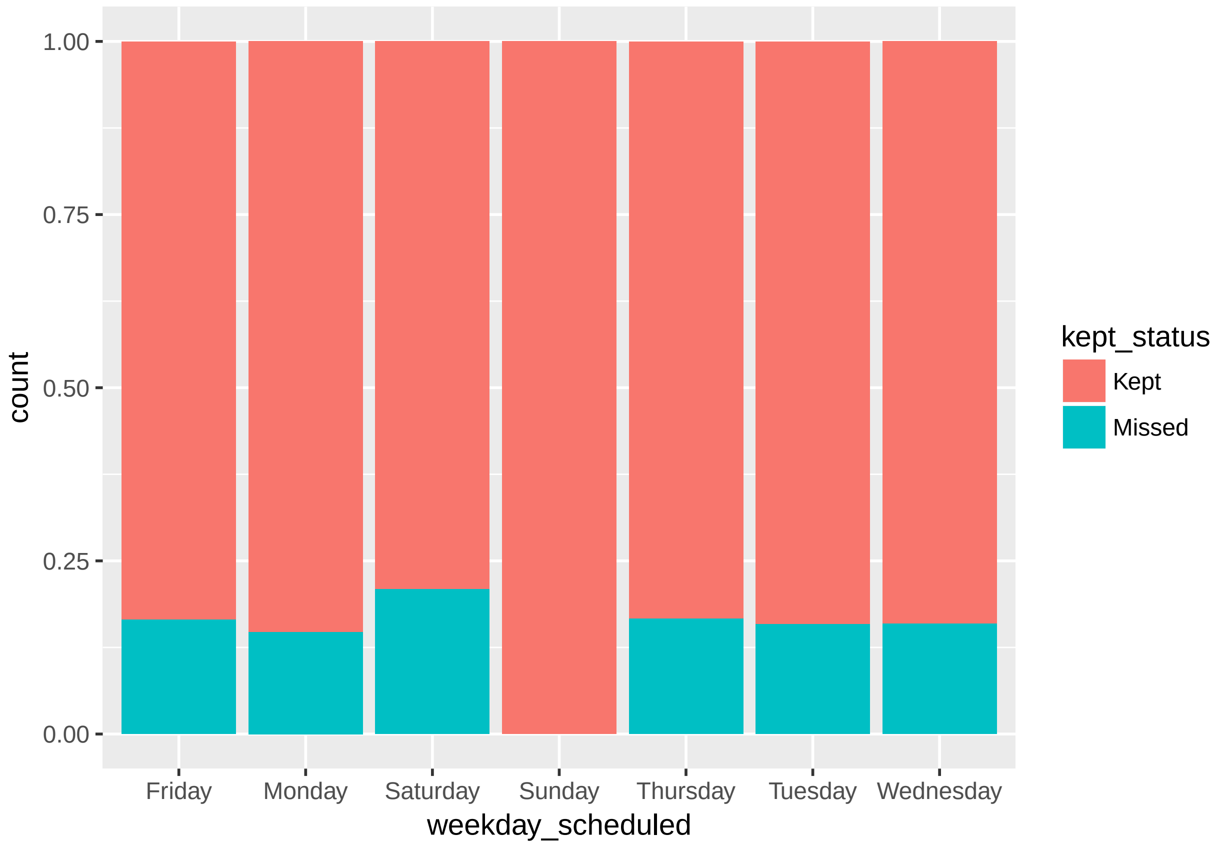
#### appt_weekday

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




**weekday_scheduled**

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




## Modeling

### Create Modeling Data

```
model_data <- appointments

factor_columns <- c(
    "kept_status", "patient_gender", "billing_type", "office_zip",
    "provider_specialty", "remind_call_result", "hour", "is_new_patient",
    "appt_weekday", "weekday_scheduled",
    "county_code")
model_data[factor_columns] <- lapply(model_data[factor_columns], factor)


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

### Divide model_data into train, validate, and test sets

The data will be dividied into three sets: train, validate, and test. I will use 60% of the tata for training, and 20% each for validation and test.

Because the data is arranged in a time-series, I want to use the most recent data for the test data, the next oldest for validation, and the oldest for training. Since I am trying to predict future appointments, testing on the most recent data will result in the best measure of the model's performance.

```
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
```

```
train_balance_subset <- train[168736:205660,]
table(train_balance_subset$kept_status)
```

```
##
##   Kept Missed
##  31060   5865
```

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

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

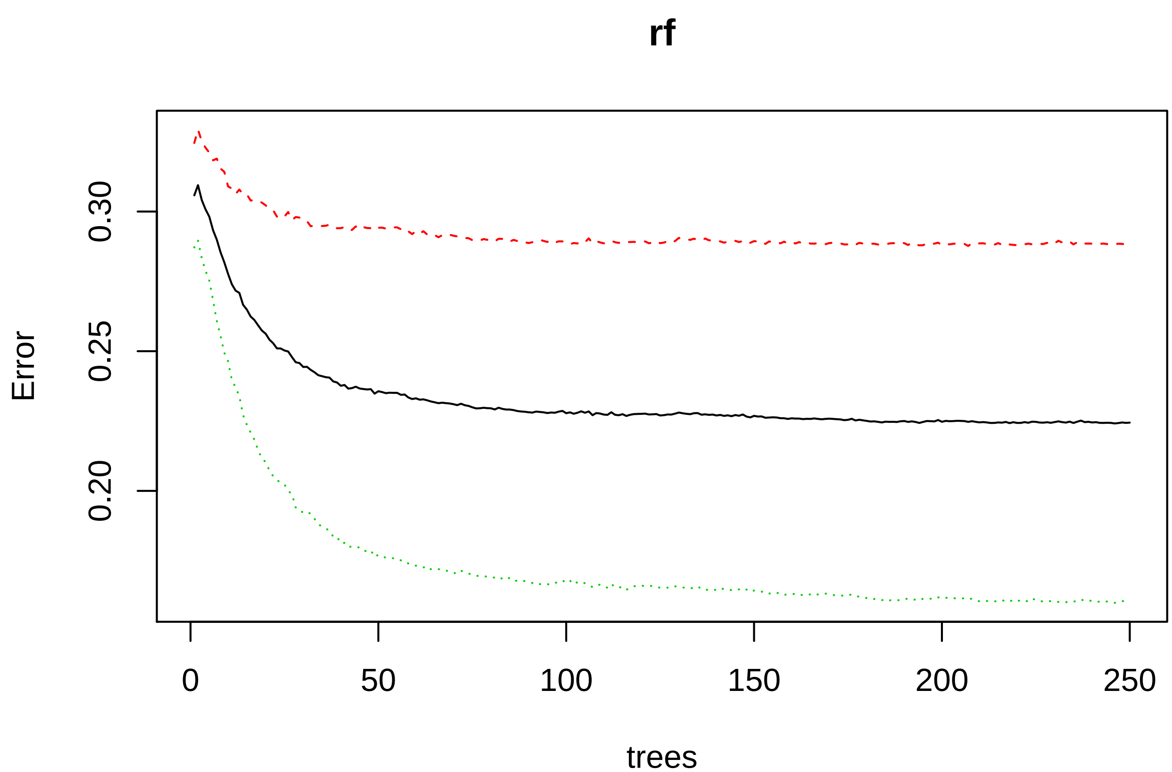
### Random Forest Model

Using randomForest Package (To be removed in final report)

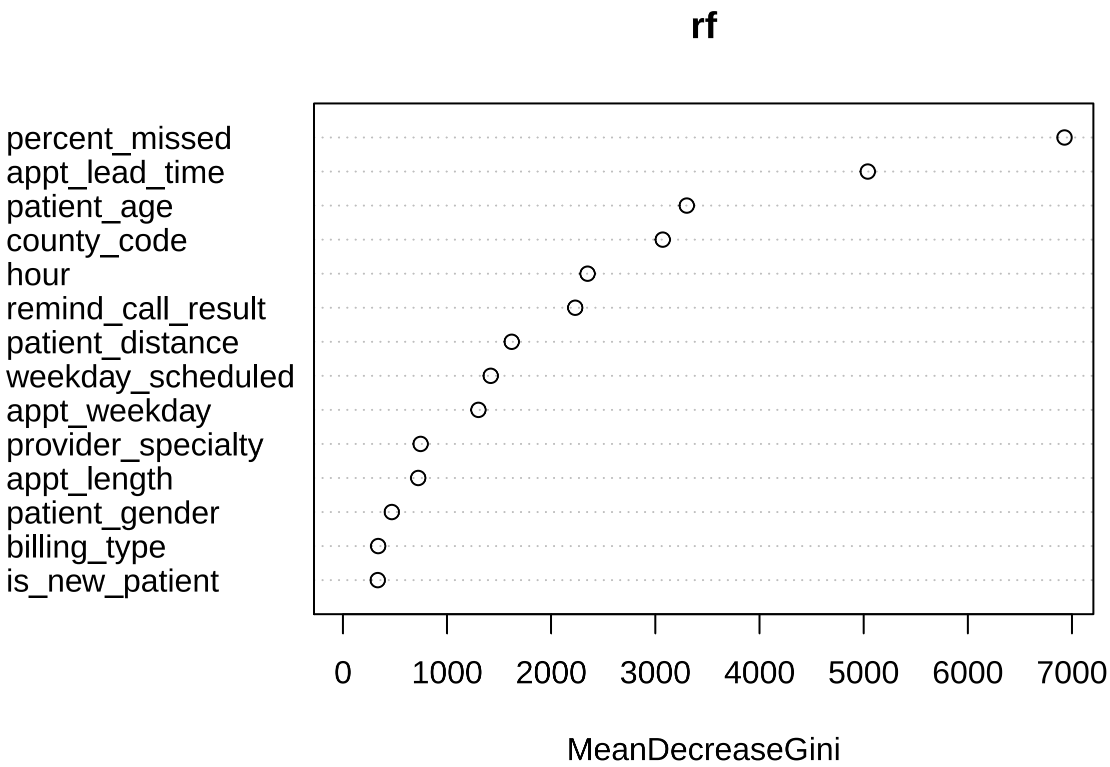
```r
rf <- randomForest(kept_status ~ ., data = train_balanced, ntree = 250)

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: 22.44%
## Confusion matrix:
##         Kept Missed class.error
## Kept   22093   8967   0.2886993
## Missed  4975  26085   0.1601739
```

```r
plot(rf)
```




```r
varImpPlot(rf)
```




Using caret Package

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

### Model Comparison

```
caret::confusionMatrix(glm_train)
pred_glm <- predict(glm_train, validate)

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

conf_mat_glm

conf_mat_glm$byClass["F1"]
```

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

caret::confusionMatrix(rftrain)

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

conf_mat_rf

conf_mat_rf$byClass["F1"]
###glm currently performing slightly better than rf on validation data based on F1 score
```