Alison Lanski
EAST
BDS Final Project: Kiva Dataset

1: Project Summary

1.1 Introduction

Context

Microlending is an appealing option to help with global development because it doesn't cost much and directly aids people in the way they want to be aided. This project is based on data provided by Kiva, a microlending company founded in 2005 which provides a platform for individuals to request and fund loans for a variety of purposes around the world. The Kiva website (www.kiva.com) provides more information about applying, lending,

Research Questions

Because Kiva is an organization operating at large scale, questions of efficiency and parity are ones that spring to mind. I am curious, for example, if there are geographic differences that play a role in the funding of requests. With many countries being served by Kiva, are applicants from some countries at an inherent advantage or disadvantage as expressed through loan-funding behaviour? Furthermore, it is possible to determine if particular elements of an application result in a request not receiving full funding? The answers to these questions could affect how Kiva markets, guides, or approves applicants to their platform and could inform their marketing techniques for lenders, assuming parity and complete full-funding are company goals.

1.2: Hypotheses

1.2.1: Hypothesis 1

If we control for the loan amount requested, predictions about funding speed will show that applicants from Africa become funded more quickly than applicants from other continents.

This hypothesis is based on the idea that, at least in America, charity and development support for Africa has a strong advertising presence generally. Therefore, it could be the case that lenders are more interested in funding projects in Africa and therefore would speed those requests towards their funding goals faster. If Kiva wants to support equal funding speed regardless of location, this analysis could show location-specific targets to market.

1.2.2: Hypothesis 2

Using a constructed sentiment score based on the "use" comments, we can predict (better than the null model) which loan requests will be fully-funded.

This hypothesis allows us to explore the predictive power of the text data that is part of the larger kiva dataset. Every funding request includes a “use” statement, which is why I have chosen that field over using the “tag” field. The overall percentage of projects that are not fully funded is small - around 7%, which means that both Kiva and requestors may be interested in factors correlated with success. As someone interested in writing, I am curious if factors relating to how the projects are written about could affect their overall funding status.

1.3: Data Description/EDA

The Kiva dataset provides information about loans requested through Kiva in the range from January 1, 2014 to July 26, 2017. With 3.5 years of data and information on 671,205 different loan requests, this dataset is a rich source of information about loan applications and funding outcomes. The dataset provides 20 fields, including amounts requested and received; dates related to funding request postings, funding amount reached, and cash disbursement; location and gender of the applicants; specifics about the purposes of the funding; and other details. We are not provided with information about loan repayment, aside from the proposed schedule. These variables default to being coded either as numeric or as text, though in practice some fields are better suited to being treated as dates or categorical variables.

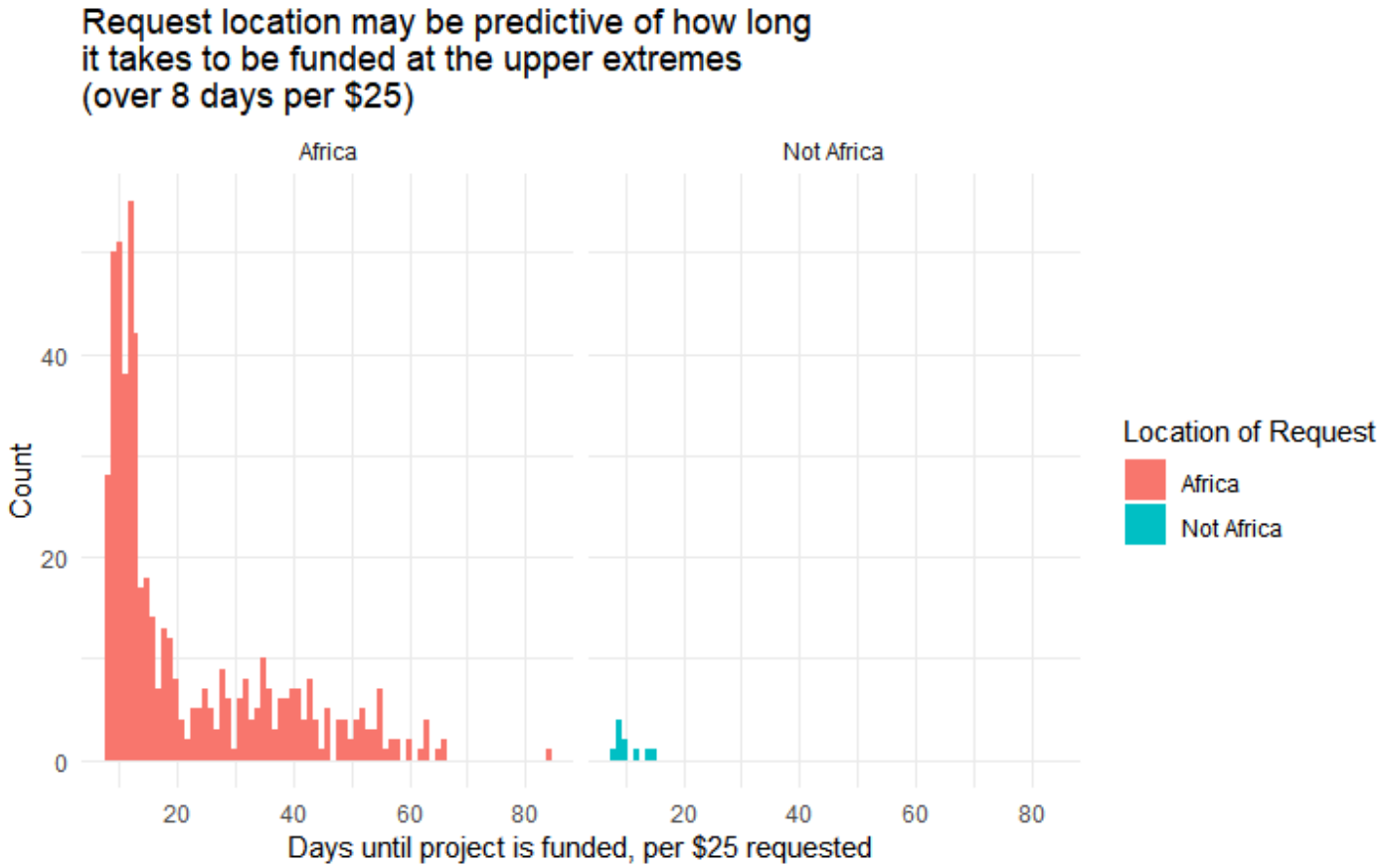
The only missing values (NAs) come from the partner_id field. These values are presumably missing because the loan applicants are not working with any partner organization, but instead apply directly to Kiva through the platform. As this missingness conveys useful information, we not exclude it from the data.

Some categorical fields have a difficult number of categories to work with. For example, the 87 countries are further divided into 12696 regions: more levels than we can reasonably interpret. The large number of categories in multiple variables dissuaded me from taking a random-forest approach to my analysis. Not only can random forest handle a limited number of categories, but many of the categories have small n which makes them difficult to guarantee in both the training and test sets.

Therefore, the second step of pre-preprocessing includes creating new fields to hold the information in the original fields in a way that is friendlier to analysis: this will include the creation of indictor or aggregate variables that can, for example, provide numeric representations of text fields. The new fields represent: the number borrowers (female, male, overall), how many tags are used in the request, how many days elapse to fund each $25 of a project on average, an indicator showing if the borrower is working with a partner organization, an indicator showing if the request receives as much as it asked for, and word counts of the “use” field

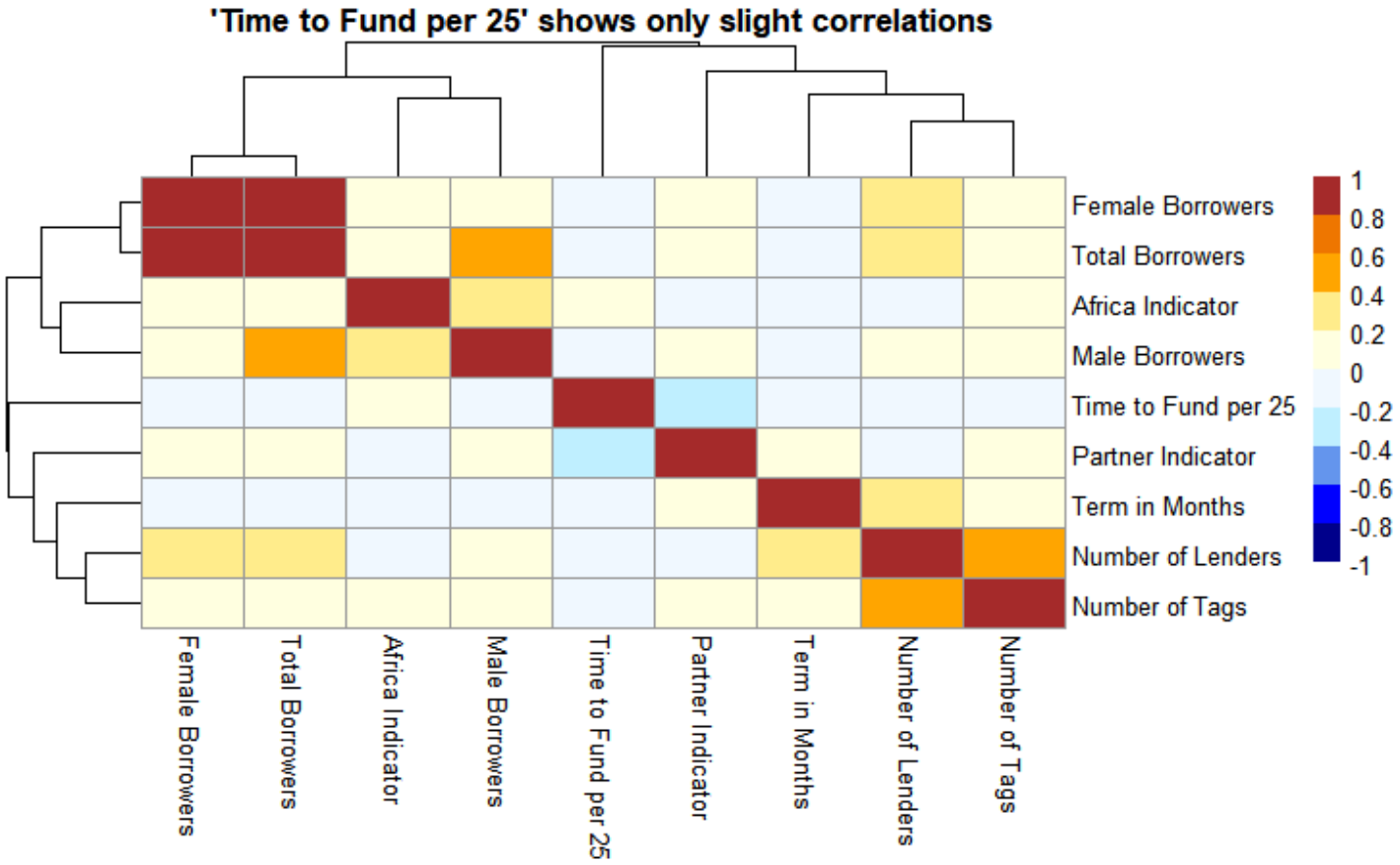
With these new variables, we can explore a few relevant relationships.

For Hypothesis 1:



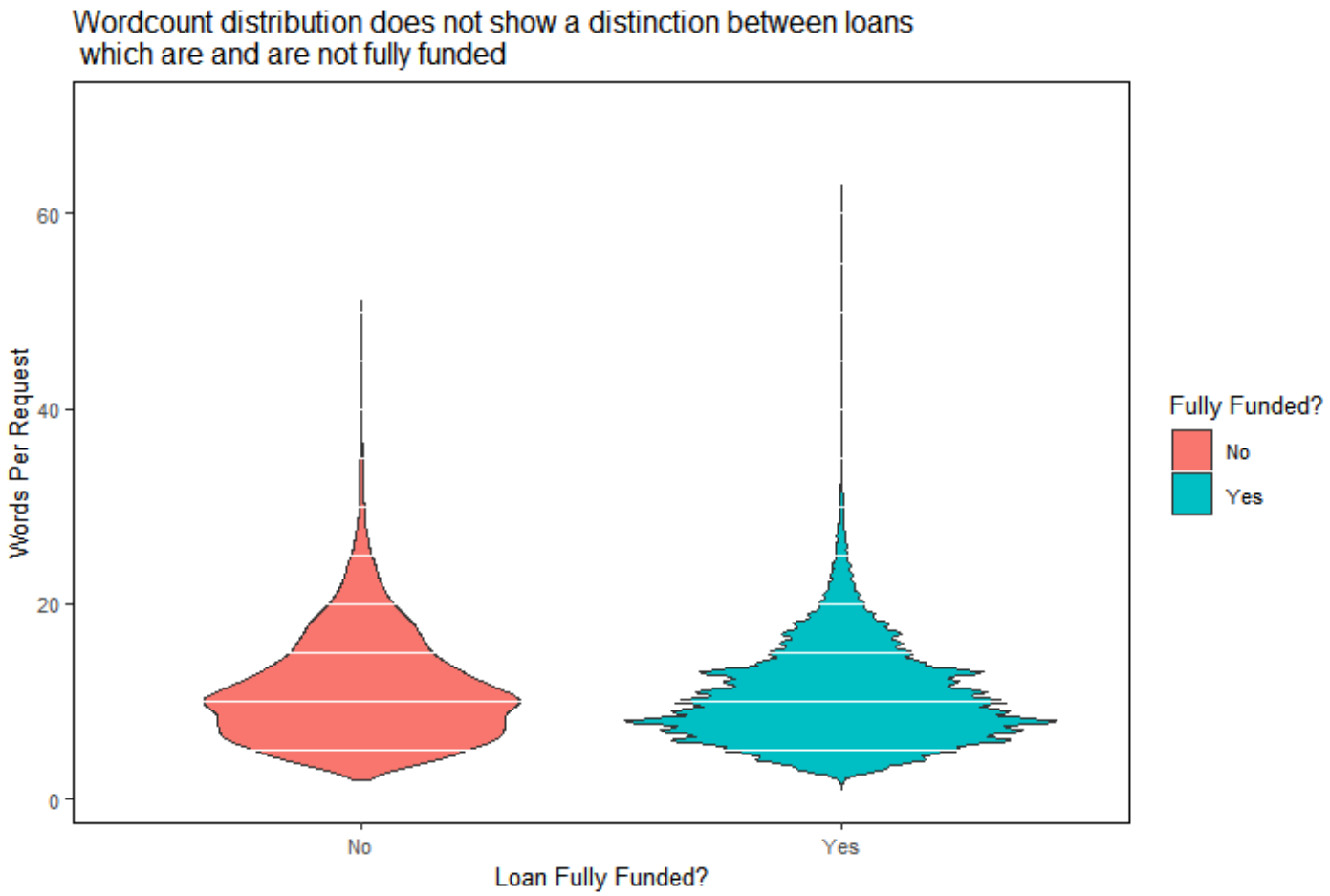
When looking at all elapsed time, the general distribution of loans is similar. But this graph suggests that at the upper extremes of funding time, being from Africa may be predictive simply because loans from other locations don't typically take this long.

Because we expect to run linear models, strong correlation between predictors muddles things and should be avoided. On the other hand, a strong correlation between our response (Time to fund per 25) and a predictor would be sign of a useful model.



The heatmap here shows that borrower numbers are strongly correlated with each other with a medium correlation bewteen number of lenders and the number of tags. Nothing is strongly correlated with the response.

In a similar vein, overall wordcount does not look like it will shed light on Hypothesis 2.



The distribution is essentially identical. The “Yes” side is less smooth than the “No” side due to a very imbalanced sample that sees most observations falling under “Yes”.

This EDA makes clear that definitively useful variables analysis may be hard to come by.

1.4: Analysis Plan

I test **Hypothesis 1** (Funding Time/Africa) by testing predictions of the calculated variable “time to fund per $25”.

1. I train several prediction models (OLS and mixed) based on variables like repayment period, sector, and a new binary field marking if the requestor is in an African country. I exclude clearly correlated fields like currency (related to country).
2. I assess the amount of variation explained by the is_africa indicator variable to see its importance as a predictor.
3. I assess the “quickness” of funding by examining the estimate (beta-hat) for my is_africa indicator, and try to determine the beta-hat’s reliability by looking at its behavior in a variety of different models.
4. Finally, I run the same models including all variables except for the is_africa variable.
5. I then predict time_to_fund_per_25 based on all models, and compare their RMSEs. If the RMSEs are similar between the models with and without the is_africa variable, we can assume that it is not an important predictor of this particular variable.

I test **Hypothesis 2** (Full Funding/Words) by evaluating classification methods used to predict the calculated indicator “fully funded”

1. I divide the training data into a traintrain and traintest set.

2. Using the traintrain data, I create three different “sentiment” scores for individual words using a subset of the training data. This first score is produced by counting the number of times each word is used and producing a proportion of how often it is found in a fully-funded “use” statement. The second score is produced by comparing the frequency of the word’s appearance out of all words used in fully-funded and non-fully-funded statements. Whichever number is larger is taken as the frequency, with the words appearing more frequently in non-fully-funded statements set as negative numbers. Finally, the sign of the frequency values are turned into a binary score variable (1 for more frequent “Yes” and 0 for more frequent “No” use).
3. I assign these word scores to the traintest words, then aggregate by id and compute an average score for the entire “use” statement. Using the overall ratio of “Yes” to “No” scores, I divide the sentence score averages by the same percentile. I can then look at the predicted assignment (via sentence score percentile) vs. the actual funding status using a confusion matrix. Comparison of confusion matrics shows which score calculation method produces the best results.
4. I apply that “best method” to the entire, unpartitioned training data to produce refined scores and the percentile split-point.
5. I then assign the recalculated word scores to the true test data, split it into “Yes” and “No” predictions based on the training percentile split-point, and use a confusion matrix to evaluate the results in comparison to the null model.

1.5: Results

Hypothesis 1

In direct contradiction to my hypothesis, every model I ran showed, with statistically significant p-values, that loan requests from Africa were likely to be funded more slowly than loan requests from elsewhere. This result was evident from the betahats associated with the models (e.g. in one OLS model, being from Not Africa had a betahat of -0.581 with $p < 2e\text{-}16$).

To make a final evaluation, I directly compared RMSE values of similar models that included and excluded an is-Africa variable.

Hypothesis 2

All confusion matrices showed that overall accuracy of predictions based on my average sentence scores was inferior to the null model: All predicted accuracies were less than 89%, with the null model's accuracy at 93%. Though all results were similar, the base-proportion model of producing word scores fared slightly better than the frequency-based scores. The best calculated score, when applied to the test data, did correctly classify 21% of the "No" values in the "fully_funded" field, unlike 0% in the null model.

1.6: Discussion

Although I produced definitive results on both hypotheses, my conclusions leave many options for further inquiry.

Hypothesis 1

For the first hypothesis, the consistency of the result feels slightly mitigated by the fact that the impact of Africa on the model was very small. For example, in the mixed model with a random slope term, the Africa Indicator accounted for only around 5% of the variation. The slight influence of the Africa Indicator was also consistent. In fact, RMSE comparisons of the models with and without the Africa indicator showed that even though the models performed better with the indicator present, they only improved by about 1% of the RMSE. In the plot above, the x-axis clearly shows the incremental improvement from red to blue dots; hardly anything with a "wow" factor.

This outcome came as a surprise to me. Much of the international charity work in the US is focused on Africa (malaria, HIV, starvation) and Kiva is an organization that has operated in the US for a long time as well. I had assumed that these factors would create additional sympathy in and for Africa as a whole, and therefore speed

funding times. Because the opposite is true, I would interested in further exploration on several measures. If the data were available, I would look at default rates and actual loan repayment periods to see if African borrowers have relatively poor repayment outcomes that might inhibit donors. Without such data, I might split the countries into more world regions to see if other areas show relatively quick or slow funding. It would also be interesting to know the nationalities or ethnicities of Kiva's funder-base. Their demographic spread might mirror the funding speed distribution we could find across diffent global regions.

Given the actual outcomes and assuming Kiva is interested in funding parity, the organization might be well served by implenting some changes to their African program, like finding a partner organization in Kenya or sponsoring more targeted advertising with an African focus.

Hypothesis 2

For the second hypothesis, I was quite disappointed that none of my word scores produced highly predictive sentence scores. The fact that none could beat the overall null model accuracy was particularly frustrating, though not entirely unexpected based on the distributions I had seen in my prework. By compelling roughly 7% of my data to be predicted as "No", as part of the model setup, it makes sense that my process would produce better Sensitivity results than the null model. Nevertheless, I had hoped that this metric would reach higher values than 21%. While this result technically satisfies my hypothesis of improving on the null model, it feels like a hollow victory. I would not consider these models in isolation to be a reliable method of predicting whether a request would be fully funded. As further inquiry, I might try to work with my scores in combination with other data, perhaps as part of a linear model or especially a random forest which could account for variable interactions. It could also be instructive to examine the test on a phrase-basis (instead of a word-basis), or to look for the effect of common gendered language markers or other sociolinguistic features.

In sum, however, if we want to determine the situation that leads to someone not receiving full funding, I would look to other methods of analysis and other variables. The ultimate goal of such an analysis for Kiva would be to use what is learned to help loan requestors produce stronger applications. If, in fact, it is difficult to point to any particular factor as a strong reason for incompete funding, these analsyes could provide moral support for applicants by assuring them their chances for obtaining full funding are very good but also essentially random.

Conclusion

This project analyzed the Kiva dataset to determine if being from Africa increased the speed of request funding, controlling for the amount requested, and if the words used in the written request can be used to predict if the loan becomes fully funded. Results indicate that African applicants have a slightly slower speed of receiving funding and that specific words are not strong predictors of receiving full project funding.

2: Appendix - Introduction to the Data

2.1: Characteristics of the Kiva Data

The Kiva dataset provides information about loans requested through Kiva in the range from January 1, 2014 to July 26, 2017. With 3.5 years of data and information on 671,205 different loan requests, this dataset is a rich source of information about loan applications and funding outcomes. The initial data provides 20 fields, including amounts requested and received; dates related to funding request postings, funding amount reached, and cash disbursement; location and gender of the applicants; specifics about the purposes of the funding; and other details.

We are not provided with information about loan repayment, aside from the proposed schedule. These variables default to being coded either as numeric or as text, though in practice some fields are better suited to being treated as dates or categorical variables.

The only missing values (NAs) in the data come from the partner_id field. These values are presumably missing because the loan applicants are not working with any partner organization, but instead apply directly to Kiva through the platform. Because the missingness of this field conveys potentially useful information, we will keep all rows of data for later analysis.

A general summary of the data follows:

```
#Load packages and data
library(tidyverse)
library(data.table)

kiva <- fread("C:/Users/Lanski/Documents/DataSets/BDS_WK11_kiva_loans.csv", header=T)

#summary to check for NAs and coltypes
summary(kiva)
```

```
##        id          funded_amount     loan_amount         activity
##  Min.   : 653047   Min.   :     0   Min.   :    25.0   Length:671205
##  1st Qu.: 823072   1st Qu.:   250   1st Qu.:   275.0   Class :character
##  Median : 992780   Median :   450   Median :   500.0   Mode  :character
##  Mean   : 993249   Mean   :   786   Mean   :   842.4
##  3rd Qu.:1163653   3rd Qu.:   900   3rd Qu.:  1000.0
##  Max.   :1340339   Max.   :100000   Max.   :100000.0
##
##     sector              use            country_code
##  Length:671205      Length:671205      Length:671205
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    country             region            currency           partner_id
##  Length:671205      Length:671205      Length:671205      Min.   :  9.0
##  Class :character   Class :character   Class :character   1st Qu.:126.0
##  Mode  :character   Mode  :character   Mode  :character   Median :145.0
##                                                           Mean   :178.2
##                                                           3rd Qu.:204.0
##                                                           Max.   :536.0
##                                                           NA's   :13507
##  posted_time        disbursed_time     funded_time        term_in_months
##  Length:671205      Length:671205      Length:671205      Min.   :  1.00
##  Class :character   Class :character   Class :character   1st Qu.:  8.00
##  Mode  :character   Mode  :character   Mode  :character   Median : 13.00
##                                                           Mean   : 13.74
##                                                           3rd Qu.: 14.00
##                                                           Max.   :158.00
##
##   lender_count         tags           borrower_genders
##  Min.   :   0.00   Length:671205      Length:671205
##  1st Qu.:   7.00   Class :character   Class :character
##  Median :  13.00   Mode  :character   Mode  :character
##  Mean   :  20.59
##  3rd Qu.:  24.00
##  Max.   :2986.00
##
##  repayment_interval     date
##  Length:671205      Length:671205
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
#summary to see what's in the data generally
glimpse(kiva)
```

```
## Observations: 671,205
## Variables: 20
## $ id                 <int> 653051, 653053, 653068, 653063, 653084, 108...
## $ funded_amount      <dbl> 300, 575, 150, 200, 400, 250, 200, 400, 475...
## $ loan_amount        <dbl> 300, 575, 150, 200, 400, 250, 200, 400, 475...
## $ activity           <chr> "Fruits & Vegetables", "Rickshaw", "Transpo...
## $ sector             <chr> "Food", "Transportation", "Transportation",...
## $ use                <chr> "To buy seasonal, fresh fruits to sell. ", ...
## $ country_code       <chr> "PK", "PK", "IN", "PK", "PK", "KE", "IN", "...
## $ country            <chr> "Pakistan", "Pakistan", "India", "Pakistan"...
## $ region             <chr> "Lahore", "Lahore", "Maynaguri", "Lahore", ...
## $ currency           <chr> "PKR", "PKR", "INR", "PKR", "PKR", "KES", "...
## $ partner_id         <dbl> 247, 247, 334, 247, 245, NA, 334, 245, 245,...
## $ posted_time        <chr> "2014-01-01 06:12:39+00:00", "2014-01-01 06...
## $ disbursed_time     <chr> "2013-12-17 08:00:00+00:00", "2013-12-17 08...
## $ funded_time        <chr> "2014-01-02 10:06:32+00:00", "2014-01-02 09...
## $ term_in_months     <dbl> 12, 11, 43, 11, 14, 4, 43, 14, 14, 11, 11, ...
## $ lender_count       <int> 12, 14, 6, 8, 16, 6, 8, 8, 19, 24, 3, 16, 1...
## $ tags               <chr> "", "", "user_favorite, user_favorite", "",...
## $ borrower_genders   <chr> "female", "female, female", "female", "fema...
## $ repayment_interval <chr> "irregular", "irregular", "bullet", "irregu...
## $ date               <chr> "2014-01-01", "2014-01-01", "2014-01-01", "...
```

2.2: Preparatory Data Processing

The first step in pre-processing is to eliminate the NAs so that our analysis methods can function appropriately. The partner_id field is presented as numbers, so we check the min and max:

```
min(kiva$partner_id, na.rm = TRUE)
```

```
## [1] 9
```

```
max(kiva$partner_id, na.rm = TRUE)
```

```
## [1] 536
```

The number 0 is available: we'll recode NA values to 0 which will indicate that there is no partner associated with this loan application.

```
#copy the dataset before manipulation
kiva2 <- kiva

#recode NA values to 0
kiva2$partner_id[is.na(kiva2$partner_id)] <- 0
```

The next step is to convert some fields into usable form. Fields may be difficult to use because they are of the wrong type (numeric instead of categorical) or contain multiple pieces of information in a single variable (e.g. borrower_genders reveals the number of applicants and their gender composition). Furthermore, most of the categorical variables exceed the number of categories that are easily handled by the analysis methods to come.

Therefore, the second step of pre-preprocessing includes creating new fields to hold the information in the original fields in a way that is friendlier to analysis: this will include data type updates and the creation of indictor or aggregate variables that can, for example, provide numeric representations of text fields. The new fields represent:
–the number of female borrowers
–the number of male borrowers
–the total number of borrowers
–how many tags are used in the request
–how many days elapse before the project is funded
–how many days elapse to fund each $25 of a project, on average
–an indicator showing if the borrower is working with a partner organization
–an indicator showing if the request receives as much as it asked for

```
###################
#borrower_genders into multiple columns
  #women
  kiva2$borrowerCount_female <- str_count(kiva2$borrower_genders, "female")

  #brute force for the men....
  kiva2$borrowerCount_male <- str_count(kiva2$borrower_genders, "male") -
                                  str_count(kiva2$borrower_genders, "female")
  #total number of borrowers
  kiva2$borrowerCount_total <- kiva2$borrowerCount_female + kiva2$borrowerCount_male

  #remove original
  kiva2 <- kiva2 %>% select(-borrower_genders)

#####################
#count how many tags are used
kiva2$number_of_tags <- str_count(kiva2$tags, ",") + #count the number of commas,
  1 - #add 1 (for the final thing)
  str_count(kiva2$tags,"^$") #subtract 1 for null entries to have them show as 0.

#####################
#how long between a request is posted and funded? (NA for never funded)
library(lubridate)
```

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

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

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

```
elapsed_time <- (kiva2$posted_time %--% kiva2$funded_time)
kiva2$time_to_fund <- (as.duration(elapsed_time)/ddays())

######################
#time to fund for every $25 in days (on average)
kiva2$time_to_fund_per_25 <- kiva2$time_to_fund/kiva2$funded_amount*25

########################
#indicator: are requests coming from a partner organization?
kiva2$with_partner <- ifelse(kiva2$partner_id == 0, "No partner", "Partner")

########################
#indicator: are requests fully funded?
kiva2$fully_funded <- ifelse(kiva2$funded_amount < kiva2$loan_amount, "No", "Yes")
```

If we examine the fields which should be treated as categorical, we find a difficult number of categories to work with. There are:

163 activities 15 sectors 87 countries and country codes 12696 regions 67 currencies 366 partners (plus “no partner”) 4 repayment intervals Other fields are expected to have unique text responses or continuous variables.

```
#number of activities?
length(unique(kiva2$activity))

#number of sectors?
length(unique(kiva2$sector))

#number of countries?
length(unique(kiva2$country))

#number of regions?
length(unique(kiva2$region))

#number of currencies?
length(unique(kiva2$currency))

#number of partners?
length(unique(kiva2$partner_id))

#number of repayment intervals?
length(unique(kiva2$repayment_interval))
```

These results show that sectors and repayment intervals have few enough categories to be practically useful; the others are so many as to exceed what our models can realistically handle, or what we can meaningfully interpret. We’ve dealt with the issue of many partners by already creating an indictor variable for “partner or no partner”

Of the other variables, some are clearly related. Region must be a subset of country, and activity must be a subset of sector. This means we can ignore region and activity, for now, in favor of the bigger bin. Country is correlated with currency, but both of those fields will have too many variables for simply analysis.

To get useable location information, we may need to zoom further out to the realm of continents. Africa is of particular interest to me, so we’ll create an “africa indicator” that shows if someone is or is not from Africa

```
#what are the countries?
sort(unique(kiva2$country))
```

```
##  [1] "Afghanistan"
##  [2] "Albania"
##  [3] "Armenia"
##  [4] "Azerbaijan"
##  [5] "Belize"
##  [6] "Benin"
##  [7] "Bhutan"
##  [8] "Bolivia"
##  [9] "Brazil"
## [10] "Burkina Faso"
## [11] "Burundi"
## [12] "Cambodia"
## [13] "Cameroon"
## [14] "Chile"
## [15] "China"
## [16] "Colombia"
## [17] "Congo"
## [18] "Costa Rica"
## [19] "Cote D'Ivoire"
## [20] "Dominican Republic"
## [21] "Ecuador"
## [22] "Egypt"
## [23] "El Salvador"
## [24] "Georgia"
## [25] "Ghana"
## [26] "Guam"
## [27] "Guatemala"
## [28] "Haiti"
## [29] "Honduras"
## [30] "India"
## [31] "Indonesia"
## [32] "Iraq"
## [33] "Israel"
## [34] "Jordan"
## [35] "Kenya"
## [36] "Kosovo"
## [37] "Kyrgyzstan"
## [38] "Lao People's Democratic Republic"
## [39] "Lebanon"
## [40] "Lesotho"
## [41] "Liberia"
## [42] "Madagascar"
## [43] "Malawi"
## [44] "Mali"
## [45] "Mauritania"
## [46] "Mexico"
## [47] "Moldova"
## [48] "Mongolia"
## [49] "Mozambique"
## [50] "Myanmar (Burma)"
## [51] "Namibia"
## [52] "Nepal"
## [53] "Nicaragua"
```

```
## [54] "Nigeria"
## [55] "Pakistan"
## [56] "Palestine"
## [57] "Panama"
## [58] "Paraguay"
## [59] "Peru"
## [60] "Philippines"
## [61] "Puerto Rico"
## [62] "Rwanda"
## [63] "Saint Vincent and the Grenadines"
## [64] "Samoa"
## [65] "Senegal"
## [66] "Sierra Leone"
## [67] "Solomon Islands"
## [68] "Somalia"
## [69] "South Africa"
## [70] "South Sudan"
## [71] "Suriname"
## [72] "Tajikistan"
## [73] "Tanzania"
## [74] "Thailand"
## [75] "The Democratic Republic of the Congo"
## [76] "Timor-Leste"
## [77] "Togo"
## [78] "Turkey"
## [79] "Uganda"
## [80] "Ukraine"
## [81] "United States"
## [82] "Vanuatu"
## [83] "Vietnam"
## [84] "Virgin Islands"
## [85] "Yemen"
## [86] "Zambia"
## [87] "Zimbabwe"
```

Interesting! Apparently our data has 87 countries compared to the 82 currently advertised. This means that our analysis won't match up 1:1 with the current countries served and therefore will be less useful if we do it in too granular of a fashion. This indicator variable will be useful now for the model but also for the end result.

```
#List of countries
africa <- c("Benin","Burkina Faso", "Burundi", "Cameroon", "Congo", "Cote D'Ivoire",
            "Egypt", "Ghana", "Kenya", "Lesotho", "Liberia", "Madagascar", "Malawi",
            "Mali", "Mauritania", "Mozambique", "Namibia", "Nigeria", "Rwanda",
            "Senegal", "Sierra Leone", "Somalia", "South Africa", "South Sudan",
            "Tanzania", "The Democratic Republic of the Congo", "Togo", "Uganda",
            "Zambia", "Zimbabwe")


kiva2$is_africa <- ifelse(kiva2$country %in% africa, "Africa", "Not Africa")
```

We now have a good base of fields with work with. We'll proceed to splitting the data into three groups: exploratory data (20%), training data (50% plus the exploratory 20%), and test data (30%).

```
#turn down scientific notation
options(scipen = 8)

#split into exploratory, training, test:  20%, 50%, 30%
rows <- nrow(kiva2)

#exploratory group: 20% of the total
set.seed(376)
index_eda <- sample(1:rows, round(0.2*rows))
kiva_eda <- kiva2[index_eda,]

#training and test group
#exclude the eda data
kiva_remaining <- kiva2[-index_eda,]
rows2 <- dim(kiva_remaining)[1]

#split into training + test; test should have 30% overall and 37.5% of remaining;
#test should have 62.5% of remaining but, when combined with the eda data, 70% overall
set.seed(5545)
index_test <- sample(1:rows2, round(0.375*rows2))
kiva_test <- kiva_remaining[index_test,]
kiva_train <- rbind(kiva_eda, kiva_remaining[-index_test,])

#clean up
rm(kiva, kiva2, kiva_remaining, index_eda, index_test, rows2, rows)
```

This division of data allows exploration on a small portion and training on a large portion, while reserving 30% for unbiased testing.

3: Appendix - Analysis

3.1: Hypothesis 1

If we control for the loan amount requested, predictions about funding speed will show that applicants from Africa become funded more quickly than applicants from other continents.

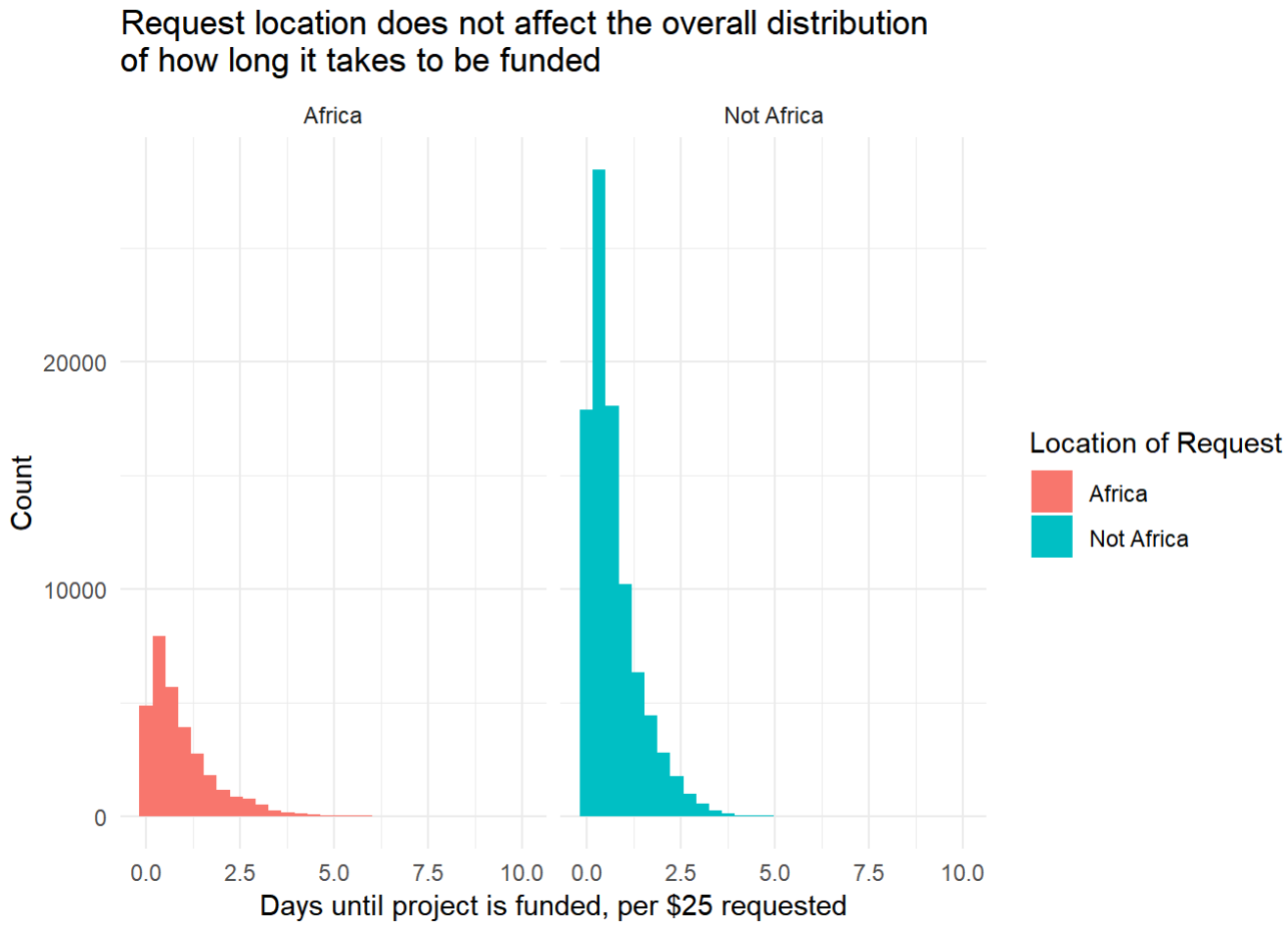
This hypothesis is based on the idea that, at least in America, charity and development support for Africa has a strong advertising presence generally. Therefore, it could be the case that lenders are more interested in funding projects in Africa and therefore would speed those requests towards their funding goals faster. If Kiva wants to support equal funding speed regardless of location, this analysis could show location-specific targets to market.

3.1.1: Exploratory Data Analysis

Let’s graph a few pictures to check correlations between location, funding speed, and other variables.

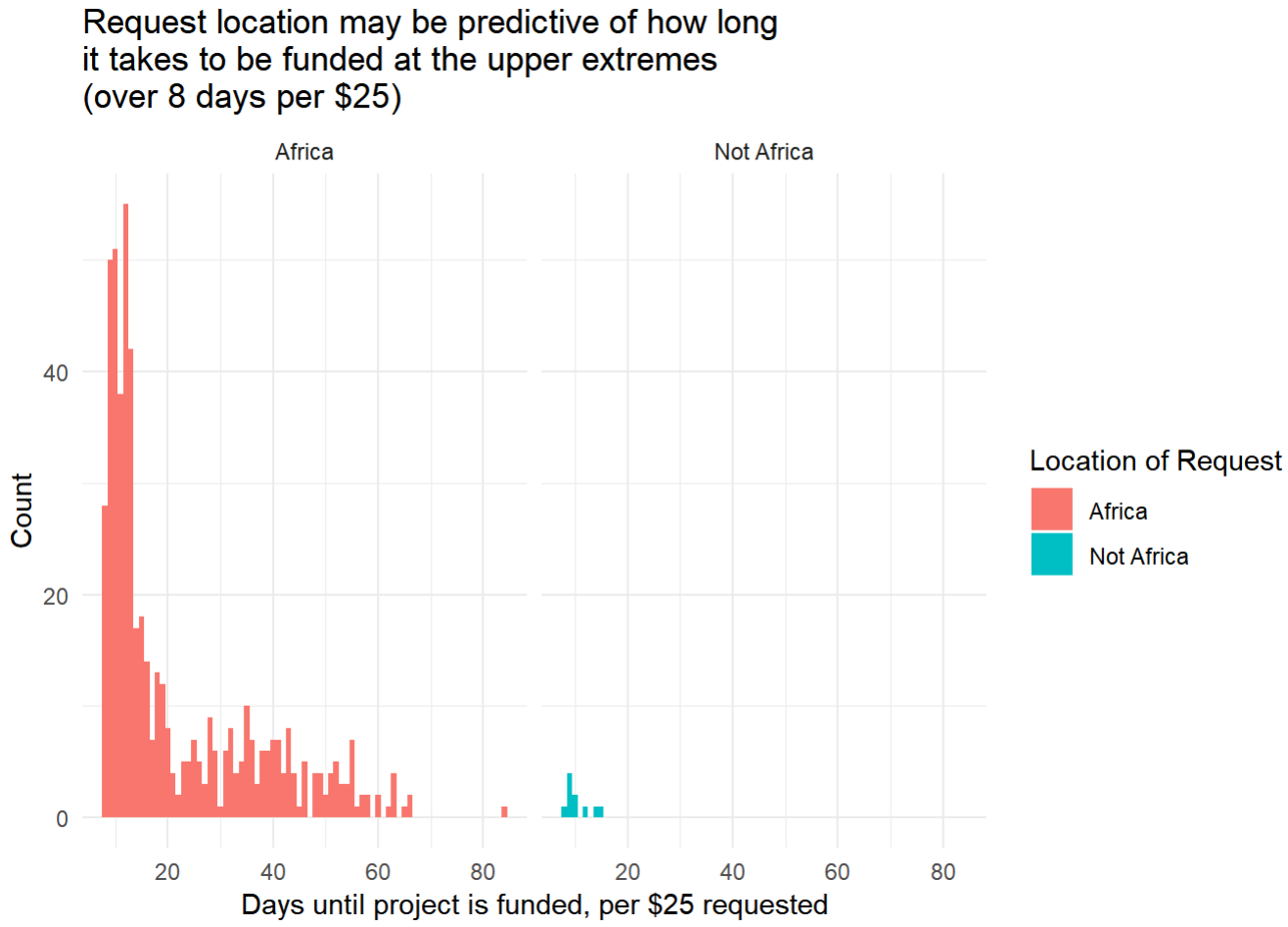
```
#distribution of time to fund

ggplot(
  data = kiva_eda[kiva_eda$time_to_fund_per_25 < 10],
  aes(x = time_to_fund_per_25, fill = is_africa)
) + geom_histogram() +
  facet_wrap(~is_africa) +
  theme_minimal() +
  ylab("Count") +
  xlab("Days until project is funded, per $25 requested") +
  scale_fill_discrete("Location of Request") +
  ggtitle("Request location does not affect the overall distribution \nof how long it takes to b
e funded")
```

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



This graph shows generally the same shape for Africa and Not Africa. But the tails are different. Let's zoom in on projects that take more than 8 days per $25 to fund.

```
ggplot(
  data = kiva_eda[kiva_eda$time_to_fund_per_25 > 8],
  aes(x = time_to_fund_per_25, fill = is_africa)
) + geom_histogram(binwidth = 1) +
  facet_wrap(~is_africa) +
  theme_minimal() +
  ylab("Count") +
  xlab("Days until project is funded, per $25 requested") +
  scale_fill_discrete("Location of Request") +
  ggtitle("Request location may be predictive of how long \nit takes to be funded at the upper e
xtremes \n(over 8 days per $25)")
```

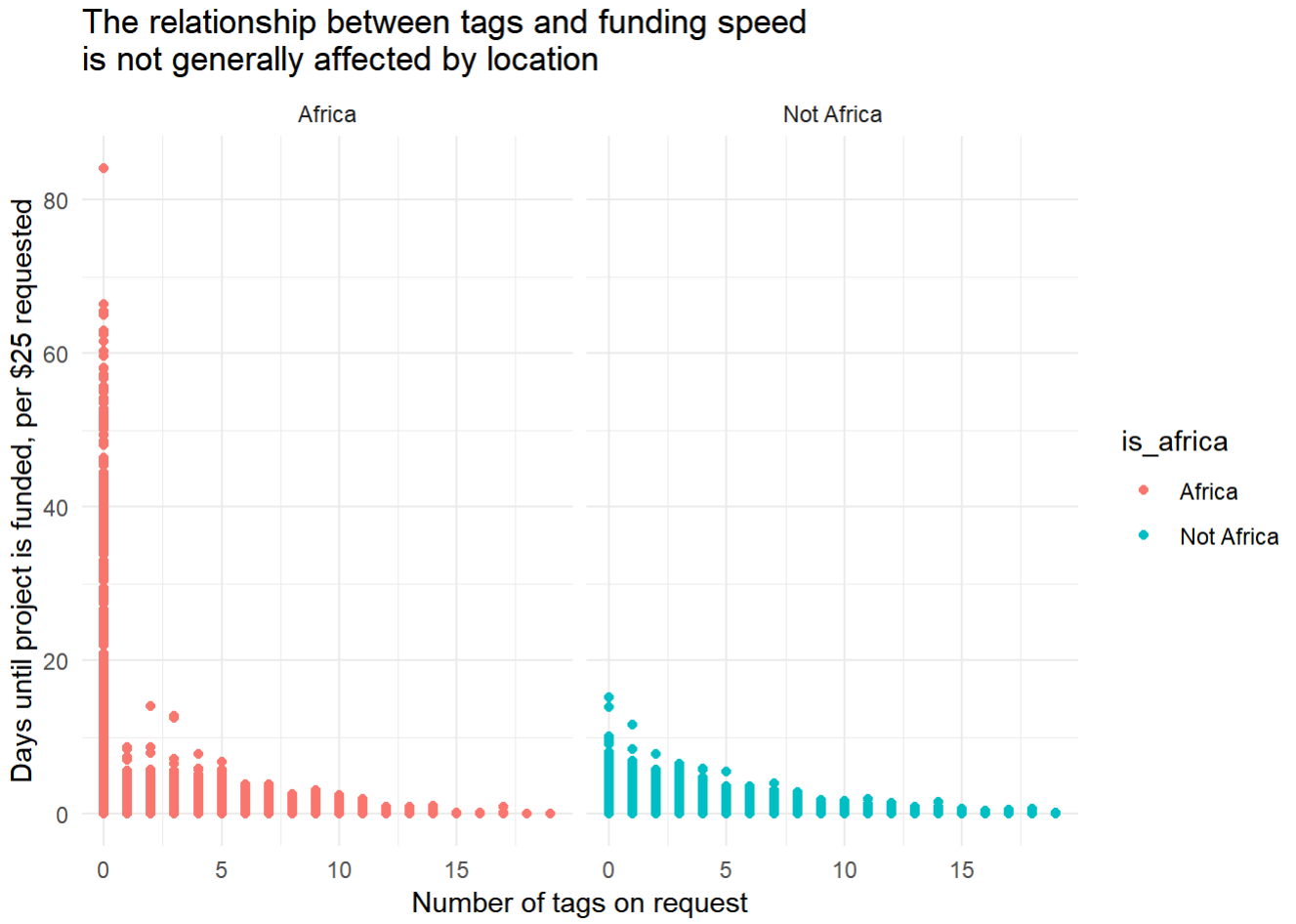


This graph suggests that at the upper extremes of funding time, being from Africa may be predictive simply because loans from other locations don't typically take this long.

Similarly, the upper extremes of funding time seem to all have no tags; but the number of tags seems otherwise unrelated to location:

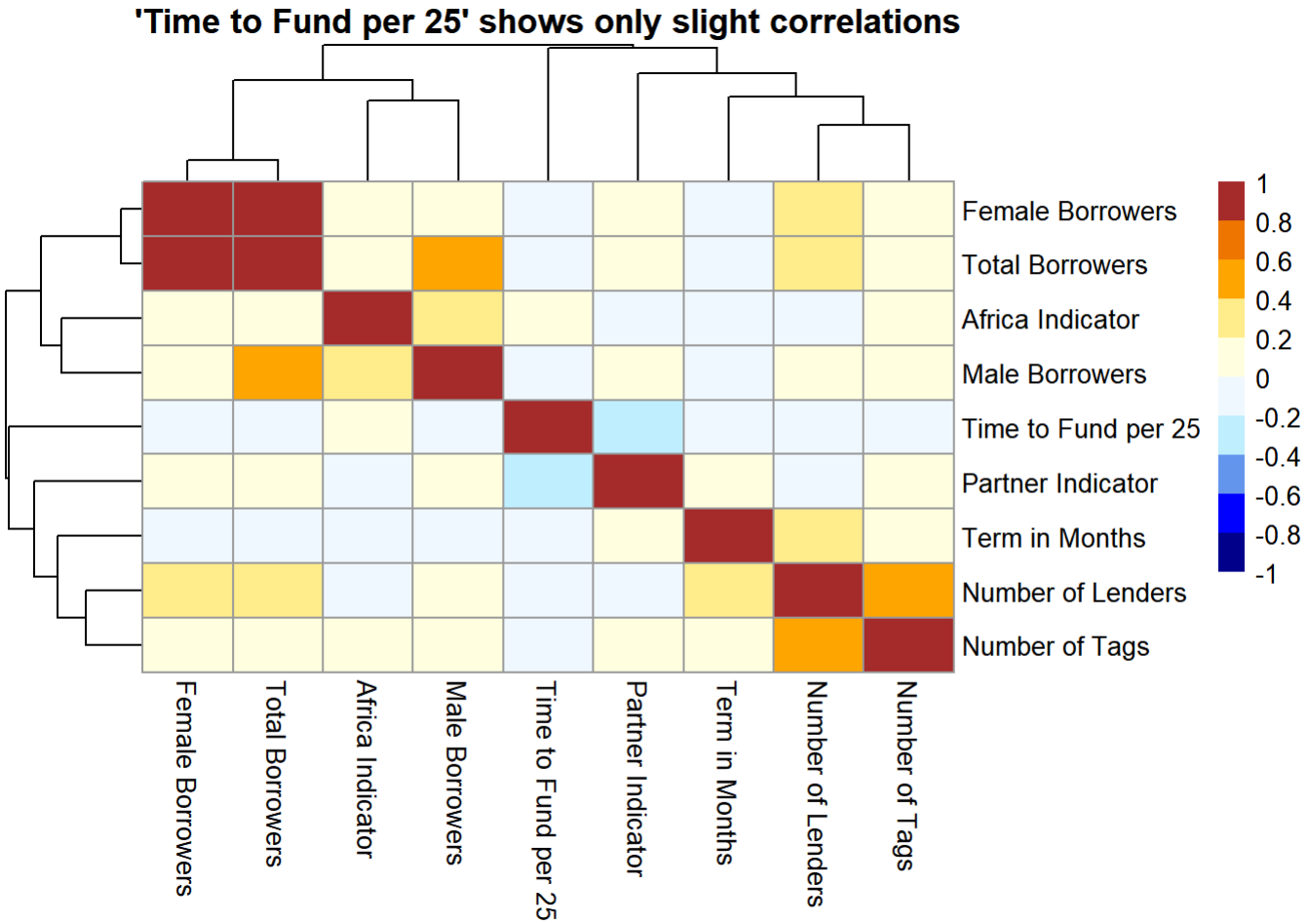
```
#how many tags on the request?
ggplot(
  data = kiva_eda[kiva_eda$number_of_tags < 20,],
  aes(x = number_of_tags, y = time_to_fund_per_25, color = is_africa)
) + geom_point() + facet_grid(~is_africa) + theme_minimal() +
  xlab("Number of tags on request") +
  ylab("Days until project is funded, per $25 requested") +
  ggtitle("The relationship between tags and funding speed \nis not generally affected by locati
on")
```

```
## Warning: Removed 9813 rows containing missing values (geom_point).
```



In preparation for linear modeling, let's examine correlations between the numeric and indicator variables, to see if any of them seem like they are well-correlated with the time to fund per $25

```
kiva_numeric <- kiva_eda %>%
  mutate(is_africa = ifelse(is_africa == "Africa", 1, 0),
         with_partner = ifelse(with_partner == "Partner", 1, 0)) %>%
           select(
                  with_partner,
                  is_africa,
                  term_in_months,
                  lender_count,
                  borrowerCount_female,
                  borrowerCount_male,
                  borrowerCount_total,
                  number_of_tags,
                  time_to_fund_per_25)

pheatmap::pheatmap(cor(kiva_numeric,
                       use = "pairwise"),
                   drop_levels = FALSE,
                   main = "'Time to Fund per 25' shows only slight correlations",
                   labels_row = c(#"Amount Funded", "Amount Requested" ,
                     "Partner Indicator", "Africa Indicator", "Term in Months",  "Number of Lend
ers",
                     "Female Borrowers", "Male Borrowers", "Total Borrowers",
                     "Number of Tags", #"Time to Fund",
                     "Time to Fund per 25"),
                   labels_col = c(#"Amount Funded", "Amount Requested" ,
                     "Partner Indicator", "Africa Indicator", "Term in Months",  "Number of Lend
ers",
                     "Female Borrowers", "Male Borrowers", "Total Borrowers",  "Number of Tags",

                     #"Time to Fund",
                     "Time to Fund per 25"),
                   breaks = seq(-1,1,0.2), legend_breaks = seq(-1,1,0.2),
                   color = c("blue4", "blue", "cornflowerblue", "lightblue1", "aliceblue",
                             "lightyellow", "lightgoldenrod1", "orange", "darkorange2", "brown"
))
```



These correlations show that the numeric and indicator variables in our data do not reveal a close relationship to the time of funding. Having a partner organization shows the strongest correlation, but only with a magnitude less than 0.4. African location does not appear to be particularly powerful, since it falls under a magnitude of 0.2, just like all remaining variables.

This EDA suggests that trying to predict the time to fund may be difficult or imprecise. Within those predictions, the Africa indicator variable is unlikely to be a strong predictor. The sign of the correlation, however, suggests that when an applicant is from Africa, their time to fund is slightly higher, in contradiction of my hypothesis.

3.1.2: Method

I will test this hypothesis by testing predictions of the calculated variable “time to fund per $25”.

I will train several prediction models based on variables like repayment period, sector, and a new binary field marking if the requestor is in an African country. I will exclude clearly correlated fields like currency (related to country).

I will assess the amount of variation explained by the is_africa indicator variable to see its importance as a predictor.

I will assess the “quickness” of funding by examining the estimate (beta-hat) for my is_africa indicator, and try to determine the beta-hat’s reliability by looking at its behavior in a variety of different models.

Finally, I will run the same models including all variables except for the is_africa variable. I will then predict time_to_fund_per_25 based on all models, and compare their RMSEs. If the RMSEs are similar between the models with and without the is_africa variable, we can assume that it is not an important predictor of this particular variable.

3.1.3: Analysis

Let's start with a base model. Because is_africa is a binary factor, it will predict time_to_fund_per_25 based on means, so let's see what those are directly,

```
kiva_train %>% group_by(is_africa) %>% summarize(avg_time_to_fund_per_25 = mean(time_to_fund_per
_25, na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##   is_africa  avg_time_to_fund_per_25
##   <chr>                        <dbl>
## 1 Africa                       1.32
## 2 Not Africa                   0.730
```

It looks like it takes almost twice as long, on average, to get funding from Africa. But we did have a large upper tail. What if we remove time to fund durations longer than 20?

```
kiva_train %>% filter(time_to_fund_per_25 <= 20) %>%
  group_by(is_africa) %>%
  summarize(avg_time_to_fund_per_25 = mean(time_to_fund_per_25, na.rm = T))
```

```
## # A tibble: 2 x 2
##   is_africa  avg_time_to_fund_per_25
##   <chr>                        <dbl>
## 1 Africa                       1.11
## 2 Not Africa                   0.730
```

Removing the extreme outliers does bring the two numbers closer, but their distance is still notable. It's not looking good for my hypothesis!

Time to start running some models.

Base model:

```
#the model
mod_h1_lm_africa <- lm(time_to_fund_per_25 ~ is_africa, dat = kiva_train)
summary(mod_h1_lm_africa)
```

```
##
## Call:
## lm(formula = time_to_fund_per_25 ~ is_africa, data = kiva_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.321 -0.591 -0.286  0.227 82.793
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.322068   0.005425   243.7   <2e-16 ***
## is_africaNot Africa -0.592049   0.006299   -94.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.82 on 435908 degrees of freedom
##   (33933 observations deleted due to missingness)
## Multiple R-squared:  0.01987,    Adjusted R-squared:  0.01986
## F-statistic:  8835 on 1 and 435908 DF,  p-value: < 2.2e-16
```

The error term is low enough and the t-value is high enough that we have statistical significance. The model's R^2 is very low though, which may show that is_africa is a significant predictor but not a strong one.

Let's see if these conclusions hold up if we add more of our numeric variables to the model. We'll need to exclude anything collinear with is_africa like: country_code, country, region, currency. Similarly, we know that funded_amount and loan_amount are very similar and highly correlated. We'll only include funded_amount in the model. I've also excluded the categorical variables, aside from "with partner" and "repayment interval", because they have a very large number of levels that will make the model difficult to interpret.

```
mod_h1_lm_more <- lm(time_to_fund_per_25 ~
                       is_africa +
                       with_partner +
                       funded_amount +
                       term_in_months +
                       lender_count +
                       repayment_interval +
                       borrowerCount_female +
                       borrowerCount_male +
                       number_of_tags,
                     dat = kiva_train)

summary(mod_h1_lm_more)
```

```
##
## Call:
## lm(formula = time_to_fund_per_25 ~ is_africa + with_partner +
##     funded_amount + term_in_months + lender_count + repayment_interval +
##     borrowerCount_female + borrowerCount_male + number_of_tags,
##     data = kiva_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.114 -0.492 -0.183  0.278 77.104
##
## Coefficients:
##                                Estimate   Std. Error t value Pr(>|t|)
## (Intercept)                 7.403885154  0.021151213  350.05  < 2e-16 ***
## is_africaNot Africa        -0.270097610  0.005924586  -45.59  < 2e-16 ***
## with_partnerPartner        -5.940575341  0.019520216 -304.33  < 2e-16 ***
## funded_amount              -0.000403196  0.000004769  -84.55  < 2e-16 ***
## term_in_months             -0.010199690  0.000319547  -31.92  < 2e-16 ***
## lender_count                0.001960266  0.000177306   11.06  < 2e-16 ***
## repayment_intervalirregular -0.302675489  0.009498631  -31.86  < 2e-16 ***
## repayment_intervalmonthly   -0.108573542  0.009003284  -12.06  < 2e-16 ***
## repayment_intervalweekly    -4.323048102  0.081386967  -53.12  < 2e-16 ***
## borrowerCount_female        0.011574574  0.000992402   11.66  < 2e-16 ***
## borrowerCount_male          -0.019591902  0.002486275   -7.88 3.28e-15 ***
## number_of_tags              0.041153478  0.001260621   32.65  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.625 on 435898 degrees of freedom
##   (33933 observations deleted due to missingness)
## Multiple R-squared:  0.2188, Adjusted R-squared:  0.2188
## F-statistic: 1.11e+04 on 11 and 435898 DF,  p-value: < 2.2e-16
```

With our expanded model, the betahat for the Africa indicator has been cut roughly in half, despite the error staying roughly the same. This change could indicate that Africa is in some way related to other predictors in the model. The high-magnitute beta-hat and t-value of the partner indicator might be a good place to look.

Let's look a little more closely at which places don't have partners.

```
kiva_train %>% filter(with_partner == "No partner") %>% group_by(country, with_partner) %>% tall
y() %>% arrange(country)
```

```
## # A tibble: 5 x 3
## # Groups:   country [5]
##   country        with_partner     n
##   <chr>          <chr>        <int>
## 1 Guam           No partner       1
## 2 Kenya          No partner    5956
## 3 Puerto Rico    No partner      45
## 4 United States  No partner    3587
## 5 Virgin Islands No partner       1
```

It seems as though partners are the norm with Kiva. Only US locations and Kenya have applicants without partners. Since Kenya and the US proper account for nearly all cases, let's take a closer look at its funding time based on the use of a partner organization

```
kiva_train %>% filter(country == "Kenya" | country == "United States") %>%
  group_by(country, with_partner) %>%
  summarize(avg_time_to_fund_per_25 = mean(time_to_fund_per_25, na.rm = TRUE))
```

```
## # A tibble: 4 x 3
## # Groups:   country [?]
##   country       with_partner avg_time_to_fund_per_25
##   <chr>         <chr>                          <dbl>
## 1 Kenya         No partner                     8.50
## 2 Kenya         Partner                        1.11
## 3 United States No partner                     0.300
## 4 United States Partner                        0.143
```

In both countries, not having a partner increases the time to fund; Kenya's increase is dramatic compared to the US. There may be underlying factors that explain why someone without a partner is funded slower, but the general situation does not seem to be unique to Africa. However, given that half of the cases without a partner are in Kenya, part of Africa, and the funding time there is dramatically longer, it could be affecting the model. Let's try again without the partner indictor.

```
mod_h1_lm_nopartner <- lm(time_to_fund_per_25 ~
                        is_africa +
                        funded_amount +
                        term_in_months +
                        lender_count +
                        repayment_interval +
                        borrowerCount_female +
                        borrowerCount_male +
                        number_of_tags,
                      dat = kiva_train)

summary(mod_h1_lm_nopartner)
```

```
##
## Call:
## lm(formula = time_to_fund_per_25 ~ is_africa + funded_amount +
##     term_in_months + lender_count + repayment_interval + borrowerCount_female +
##     borrowerCount_male + number_of_tags, data = kiva_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.926 -0.577 -0.237  0.242 82.644
##
## Coefficients:
##                                Estimate   Std. Error t value Pr(>|t|)
## (Intercept)                 1.918812842  0.012188593 157.427  < 2e-16 ***
## is_africaNot Africa        -0.581047979  0.006425944 -90.422  < 2e-16 ***
## funded_amount              -0.000175996  0.000005186 -33.937  < 2e-16 ***
## term_in_months             -0.015227270  0.000351390 -43.334  < 2e-16 ***
## lender_count               -0.000277878  0.000195067  -1.425  0.15430
## repayment_intervalirregular -0.302050125  0.010459152 -28.879  < 2e-16 ***
## repayment_intervalmonthly  -0.094172107  0.009913577  -9.499  < 2e-16 ***
## repayment_intervalweekly    1.207325129  0.087354485  13.821  < 2e-16 ***
## borrowerCount_female       -0.031715249  0.001081473 -29.326  < 2e-16 ***
## borrowerCount_male         -0.068298039  0.002732015 -24.999  < 2e-16 ***
## number_of_tags             -0.004096910  0.001378408  -2.972  0.00296 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.789 on 435899 degrees of freedom
##   (33933 observations deleted due to missingness)
## Multiple R-squared:  0.05285,    Adjusted R-squared:  0.05283
## F-statistic:  2432 on 10 and 435899 DF,  p-value: < 2.2e-16
```

This looks much better. The betahat, standard error, and t-value for the Africa Indicator are very similar to the base model. By removing the partner indicator, we arrive at a model whose consistency with the base model suggests that the Africa beta-hat is probably not related to the other predictors in the model and is representing a separate piece of information. However, the fact that the R2 is much better with these other features suggests that even though we may have the right beta-hat, it's not a strong contributer to explaining the variation of our time_to_fund_per_25. The other things we added (or some subset of them) appear to be contributing much more to our understanding of the variation in time_to_fund_per_25.

Let's try a mixed model with is_africa as a random effect. We'll try it as a random intercept (which should produce similar results) and as a random slope.

```
library(lme4)
```

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

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

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

```
mod_h1_mm_int <- lmer(time_to_fund_per_25 ~
                        funded_amount +
                        term_in_months +
                        lender_count +
                        repayment_interval +
                        borrowerCount_female +
                        borrowerCount_male +
                        number_of_tags +
                        (1|is_africa),
                      data = kiva_train)
```

```
## Warning: Some predictor variables are on very different scales: consider
## rescaling
```

```
summary(mod_h1_mm_int)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## time_to_fund_per_25 ~ funded_amount + term_in_months + lender_count +
##     repayment_interval + borrowerCount_female + borrowerCount_male +
##     number_of_tags + (1 | is_africa)
##    Data: kiva_train
##
## REML criterion at convergence: 1744395
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.635 -0.323 -0.133  0.135 46.189
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  is_africa (Intercept) 0.1688   0.4108
##  Residual              3.2014   1.7892
## Number of obs: 435910, groups:  is_africa, 2
##
## Fixed effects:
##                                 Estimate   Std. Error t value
## (Intercept)                  1.628278702  0.290735177   5.601
## funded_amount               -0.000176003  0.000005186 -33.938
## term_in_months              -0.015227631  0.000351389 -43.335
## lender_count                -0.000277737  0.000195067  -1.424
## repayment_intervalirregular -0.302057347  0.010459149 -28.880
## repayment_intervalmonthly   -0.094175814  0.009913577  -9.500
## repayment_intervalweekly     1.207370189  0.087354473  13.822
## borrowerCount_female        -0.031713859  0.001081472 -29.325
## borrowerCount_male          -0.068293229  0.002732010 -24.997
## number_of_tags              -0.004096767  0.001378408  -2.972
##
## Correlation of Fixed Effects:
##               (Intr) fndd_m trm_n_ lndr_c rpymnt_ntrvlr rpymnt_ntrvlm
## funded_amnt    0.004
## trm_n_mnths   -0.020 -0.035
## lender_cont   -0.001 -0.820 -0.099
## rpymnt_ntrvlr -0.033 -0.101  0.224  0.056
## rpymnt_ntrvlm -0.031 -0.080  0.113  0.011  0.824
## rpymnt_ntrvlw -0.005  0.002  0.046 -0.017  0.103         0.100
## brrwrCnt_fm   -0.008 -0.483  0.184  0.260  0.039         0.078
## brrwrCnt_ml   -0.012 -0.080  0.092 -0.001  0.264         0.238
## numbr_f_tgs   -0.008  0.024 -0.065 -0.235  0.069         0.008
##               rpymnt_ntrvlw brrwrCnt_f brrwrCnt_m
## funded_amnt
## trm_n_mnths
## lender_cont
## rpymnt_ntrvlr
## rpymnt_ntrvlm
## rpymnt_ntrvlw
## brrwrCnt_fm    0.014
## brrwrCnt_ml    0.043        -0.051
## numbr_f_tgs    0.035        -0.016     -0.070
```

```
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

The beta-hats of our other predictors remain pretty steady, as do the standard errors, under the mixed model with random intercept. The is_africa random intercept term explains only ~3% of the variation contained in all random effects, again showing that its significance is not large.

As expected, using this term as a random intercept is virtually identical to having it as an element in the model, because it's simply a binary indicator. Let's look at a random slope model instead. In order to run this model, I need a hierarchical model, so I'll look at with respect to a random intercept of repayment interval (which only has three levels)

```
mod_h1_mm_slp <- lmer(time_to_fund_per_25 ~
                        funded_amount +
                        term_in_months +
                        lender_count +
                        repayment_interval +
                        borrowerCount_female +
                        borrowerCount_male +
                        number_of_tags +
                        (1 + is_africa|repayment_interval),
                      data = kiva_train)
```

```
## Warning: Some predictor variables are on very different scales: consider
## rescaling
```

```
summary(mod_h1_mm_slp)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## time_to_fund_per_25 ~ funded_amount + term_in_months + lender_count +
##     repayment_interval + borrowerCount_female + borrowerCount_male +
##     number_of_tags + (1 + is_africa | repayment_interval)
##    Data: kiva_train
##
## REML criterion at convergence: 1743968
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.633 -0.322 -0.132  0.136 46.289
##
## Random effects:
##  Groups             Name                Variance Std.Dev. Corr
##  repayment_interval (Intercept)         4.9129   2.2165
##                     is_africaNot Africa 0.3484   0.5903   -0.22
##  Residual                               3.1982   1.7883
## Number of obs: 435910, groups:  repayment_interval, 4
##
## Fixed effects:
##                                 Estimate   Std. Error t value
## (Intercept)                  1.404028086  2.160617116   0.650
## funded_amount               -0.000178936  0.000005187 -34.494
## term_in_months              -0.014596368  0.000352549 -41.402
## lender_count                -0.000039995  0.000195300  -0.205
## repayment_intervalirregular -0.273769877  3.055548203  -0.090
## repayment_intervalmonthly   -0.092515734  3.055547457  -0.030
## repayment_intervalweekly     1.715468422  3.096547800   0.554
## borrowerCount_female        -0.030724082  0.001083837 -28.348
## borrowerCount_male          -0.066905269  0.002767122 -24.179
## number_of_tags              -0.003960227  0.001380876  -2.868
##
## Correlation of Fixed Effects:
##               (Intr) fndd_m trm_n_ lndr_c rpymnt_ntrvlr rpymnt_ntrvlm
## funded_amnt    0.000
## trm_n_mnths   -0.003 -0.036
## lender_cont    0.000 -0.820 -0.093
## rpymnt_ntrvlr -0.707  0.000  0.001  0.000
## rpymnt_ntrvlm -0.707  0.000  0.000  0.000  0.500
## rpymnt_ntrvlw -0.698  0.000  0.001 -0.001  0.493         0.493
## brrwrCnt_fm   -0.001 -0.484  0.185  0.261  0.000         0.000
## brrwrCnt_ml   -0.001 -0.085  0.090 -0.001  0.001         0.001
## numbr_f_tgs   -0.001  0.022 -0.066 -0.235  0.000         0.000
##               rpymnt_ntrvlw brrwrCnt_f brrwrCnt_m
## funded_amnt
## trm_n_mnths
## lender_cont
## rpymnt_ntrvlr
## rpymnt_ntrvlm
## rpymnt_ntrvlw
## brrwrCnt_fm    0.000
## brrwrCnt_ml    0.001        -0.040
```

```
## numbr_f_tgs    0.001        -0.011     -0.058
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

The results show that repayment interval has a much stronger random effect than is_africa when used as a random slope, However is_africa now accounts for roughly 5% of the variation.

So it contributes a little bit more in the random slope model than the random intercept model, although in either case its contribution in negligable compared to the other random effects.

With verions of linear models all showing that is_africa is a statistically significant but not a strong predictor, let's run the corrollary models (without is_africa) so we can compare errors.
The OLS models are straightfoward to run:

```
mod_h1_lm_noafrica <- lm(time_to_fund_per_25 ~ 1, data = kiva_train)

mod_h1_lm_more_noafrica <- lm(time_to_fund_per_25 ~
                        funded_amount +
                        term_in_months +
                        lender_count +
                        repayment_interval +
                        borrowerCount_female +
                        borrowerCount_male +
                        number_of_tags,
                      dat = kiva_train)

summary(mod_h1_lm_noafrica)
```

```
##
## Call:
## lm(formula = time_to_fund_per_25 ~ 1, data = kiva_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.883 -0.644 -0.333  0.208 83.232
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.882920   0.002785   317.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.838 on 435909 degrees of freedom
##   (33933 observations deleted due to missingness)
```

```
summary(mod_h1_lm_more_noafrica)
```

```
##
## Call:
## lm(formula = time_to_fund_per_25 ~ funded_amount + term_in_months +
##     lender_count + repayment_interval + borrowerCount_female +
##     borrowerCount_male + number_of_tags, data = kiva_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.921 -0.607 -0.272  0.211 83.084
##
## Coefficients:
##                                Estimate   Std. Error t value   Pr(>|t|)
## (Intercept)                 1.545297204  0.011574305 133.511    < 2e-16
## funded_amount              -0.000226166  0.000005204 -43.457    < 2e-16
## term_in_months             -0.018175871  0.000353139 -51.469    < 2e-16
## lender_count                0.000876010  0.000196466   4.459 0.00000824
## repayment_intervalirregular -0.361101747  0.010536176 -34.273    < 2e-16
## repayment_intervalmonthly   -0.124478484  0.010000388 -12.447    < 2e-16
## repayment_intervalweekly     1.575746064  0.088073872  17.891    < 2e-16
## borrowerCount_female        -0.020354669  0.001084176 -18.774    < 2e-16
## borrowerCount_male          -0.028967535  0.002722342 -10.641    < 2e-16
## number_of_tags              -0.002928105  0.001391213  -2.105     0.0353
##
## (Intercept)                 ***
## funded_amount               ***
## term_in_months              ***
## lender_count                ***
## repayment_intervalirregular ***
## repayment_intervalmonthly   ***
## repayment_intervalweekly    ***
## borrowerCount_female        ***
## borrowerCount_male          ***
## number_of_tags              *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.806 on 435900 degrees of freedom
##   (33933 observations deleted due to missingness)
## Multiple R-squared:  0.03509,    Adjusted R-squared:  0.03507
## F-statistic:  1761 on 9 and 435900 DF,  p-value: < 2.2e-16
```

```
anova(mod_h1_lm_more_noafrica, mod_h1_lm_more)
```

```
## Analysis of Variance Table
##
## Model 1: time_to_fund_per_25 ~ funded_amount + term_in_months + lender_count +
##     repayment_interval + borrowerCount_female + borrowerCount_male +
##     number_of_tags
## Model 2: time_to_fund_per_25 ~ is_africa + with_partner + funded_amount +
##     term_in_months + lender_count + repayment_interval + borrowerCount_female +
##     borrowerCount_male + number_of_tags
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1 435900 1421667
## 2 435898 1150947  2    270720 51265 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The mixed models don't have a clear companion model; however because I added the repayment_interval as an random intercept to create a random slope model, I'll run a verion with that as a random intercept only.

```
mod_h1_mm_int_noafrica <- lmer(time_to_fund_per_25 ~
                           funded_amount +
                           term_in_months +
                           lender_count +
                           borrowerCount_female +
                           borrowerCount_male +
                           number_of_tags +
                           (1|repayment_interval),
                         data = kiva_train)
```

```
## Warning: Some predictor variables are on very different scales: consider
## rescaling
```

```
summary(mod_h1_mm_int_noafrica)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## time_to_fund_per_25 ~ funded_amount + term_in_months + lender_count +
##     borrowerCount_female + borrowerCount_male + number_of_tags +
##     (1 | repayment_interval)
##    Data: kiva_train
##
## REML criterion at convergence: 1752495
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -1.610 -0.336 -0.151  0.117 46.006
##
## Random effects:
##  Groups             Name        Variance Std.Dev.
##  repayment_interval (Intercept) 0.768    0.8763
##  Residual                       3.261    1.8059
## Number of obs: 435910, groups:  repayment_interval, 4
##
## Fixed effects:
##                            Estimate   Std. Error t value
## (Intercept)           1.814664598  0.438742047   4.136
## funded_amount        -0.000226178  0.000005204 -43.459
## term_in_months       -0.018177651  0.000353136 -51.475
## lender_count          0.000876708  0.000196466   4.462
## borrowerCount_female -0.020355185  0.001084175 -18.775
## borrowerCount_male   -0.028972098  0.002722330 -10.642
## number_of_tags       -0.002934422  0.001391207  -2.109
##
## Correlation of Fixed Effects:
##             (Intr) fndd_m trm_n_ lndr_c brrwrCnt_f brrwrCnt_m
## funded_amnt  0.002
## trm_n_mnths -0.009 -0.045
## lender_cont -0.001 -0.820 -0.093
## brrwrCnt_fm -0.004 -0.477  0.196  0.254
## brrwrCnt_ml -0.003 -0.064  0.109 -0.012 -0.071
## numbr_f_tgs -0.003  0.025 -0.065 -0.237 -0.017     -0.073
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
```

With these models now created, we can use all versions to predict. (We exlude the “more” linear model including partner, since the correlation with Africa is too high for good comparisons)

```
#compute predictions for all models and display quantiles

pred_h1_lm_africa <- predict(object = mod_h1_lm_africa,
                             newdata = kiva_test)
summary(pred_h1_lm_africa)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7300  0.7300  0.7300  0.8824  1.3221  1.3221
```

```
#pred_h1_lm_more <- predict(object = mod_h1_lm_more,
#                           newdata = kiva_test)
#summary(pred_h1_lm_more)


pred_h1_lm_nopartner <- predict(object = mod_h1_lm_nopartner,
                                newdata = kiva_test)
summary(pred_h1_lm_nopartner)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.6950   0.7296   0.8429   0.8823   1.0323   3.0547
```

```
pred_h1_mm_int <- predict(object = mod_h1_mm_int,
                          newdata = kiva_test)
summary(pred_h1_mm_int)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.6952   0.7297   0.8429   0.8823   1.0323   3.0547
```

```
pred_h1_mm_slp <- predict(object = mod_h1_mm_slp,
                          newdata = kiva_test)
summary(pred_h1_mm_slp)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.5709   0.7370   0.8557   0.8819   1.0027   3.0514
```

```
pred_h1_lm_noafrica <- predict(object = mod_h1_lm_noafrica,
                               newdata = kiva_test)
summary(pred_h1_lm_noafrica)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8829  0.8829  0.8829  0.8829  0.8829  0.8829
```

```
pred_h1_lm_more_noafrica <- predict(object = mod_h1_lm_more_noafrica,
                                    newdata = kiva_test)
summary(pred_h1_lm_more_noafrica)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -12.4980   0.8291   0.9497   0.8832   1.0535   3.0621
```

```
pred_h1_mm_int_noafrica <- predict(object = mod_h1_mm_int_noafrica,
                                   newdata = kiva_test)
summary(pred_h1_mm_int_noafrica)
```

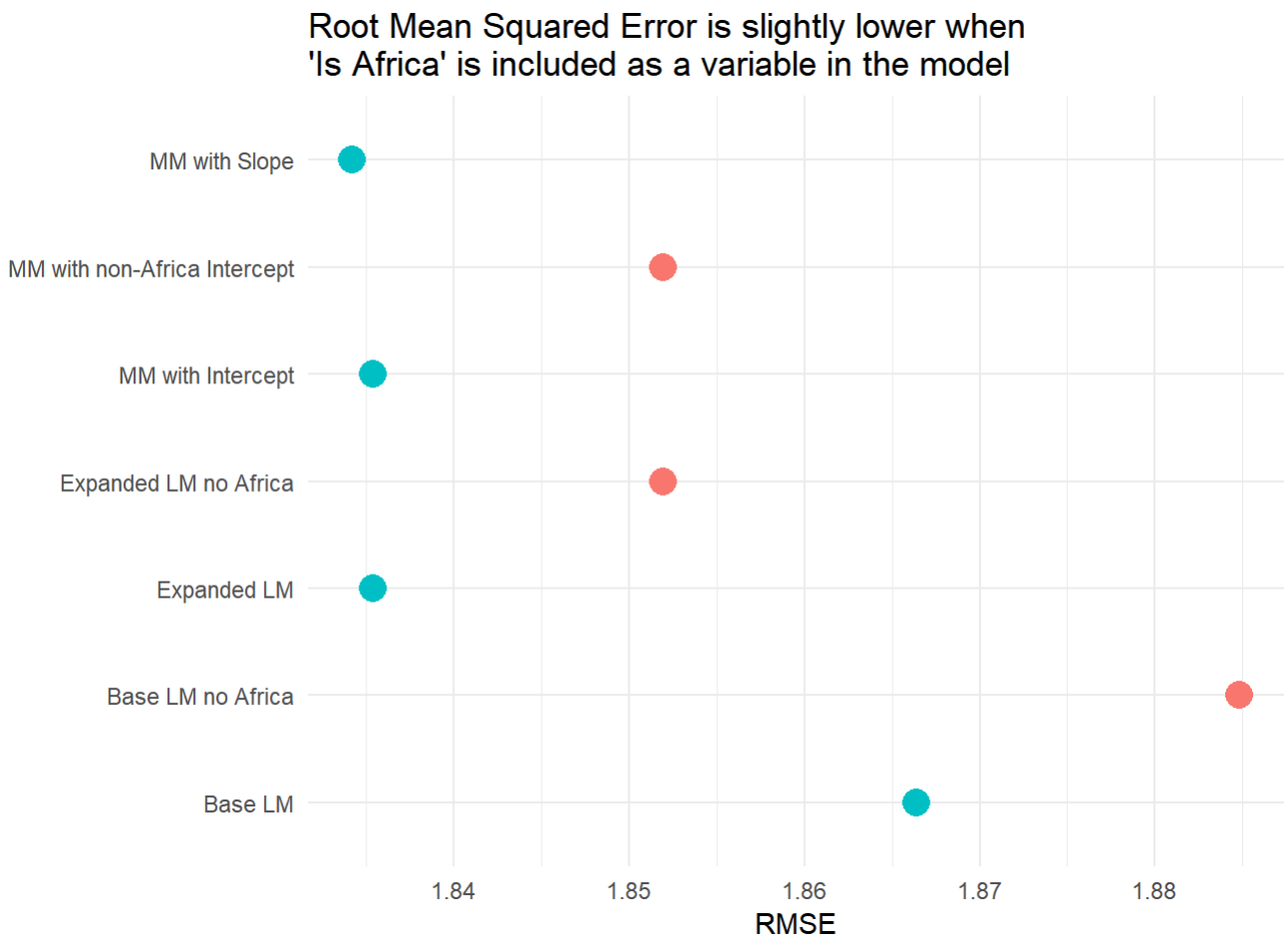
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -12.4986   0.8291   0.9497   0.8832   1.0536   3.0493
```

The model using basic OLS and a lot of variables has the most extreme values compared. The other models using more than just a default intercept have similar ranges of alues, with consistency bewteen the groups that include the Africa variable and those that don't. This suggests that the presence of Africa status does affect the model to some extent. To see more, let's calculate and compare RMSEs.

```
#compute RMSE of all models
RMSE_h1_lm_africa <-  sqrt(mean((pred_h1_lm_africa - kiva_test$time_to_fund_per_25)^2, na.rm = T
RUE))
#RMSE_h1_lm_more <-  sqrt(mean((pred_h1_lm_more - kiva_test$time_to_fund_per_25)^2, na.rm = TRU
E))
RMSE_h1_lm_nopartner <- sqrt(mean((pred_h1_lm_nopartner - kiva_test$time_to_fund_per_25)^2, na.r
m = TRUE))
RMSE_h1_mm_int <-  sqrt(mean((pred_h1_mm_int - kiva_test$time_to_fund_per_25)^2, na.rm = TRUE))
RMSE_h1_mm_slp <-  sqrt(mean((pred_h1_mm_slp - kiva_test$time_to_fund_per_25)^2, na.rm = TRUE))
RMSE_h1_lm_noafrica <-  sqrt(mean((pred_h1_lm_noafrica - kiva_test$time_to_fund_per_25)^2, na.rm
 = TRUE))
RMSE_h1_lm_more_noafrica <-  sqrt(mean((pred_h1_lm_more_noafrica - kiva_test$time_to_fund_per_2
5)^2, na.rm = TRUE))
RMSE_h1_mm_int_noafrica <-  sqrt(mean((pred_h1_mm_int_noafrica - kiva_test$time_to_fund_per_25)^
2, na.rm = TRUE))
```

Now we can tidy the results and compare in a graph

```
#combine RMSE results
h1_RMSE <- as.data.frame(rbind(RMSE_h1_lm_africa, #RMSE_h1_lm_more,
                               RMSE_h1_lm_nopartner,
                               RMSE_h1_lm_more_noafrica, RMSE_h1_lm_noafrica,
                               RMSE_h1_mm_int, RMSE_h1_mm_int_noafrica,
                               RMSE_h1_mm_slp))
colnames(h1_RMSE) <- "RMSE"
model_info <- c("Base LM", "Expanded LM",
                "Expanded LM no Africa",
                "Base LM no Africa", "MM with Intercept",
                "MM with non-Africa Intercept", "MM with Slope")

h1_RMSE <- as.data.frame(cbind(model_info, h1_RMSE))
h1_RMSE$model_info <- as.character(h1_RMSE$model_info)

ggplot(
  data = h1_RMSE,
  aes(x = RMSE, y = model_info, color = ifelse(str_detect(h1_RMSE$model_info, "Africa"), "blue",
 "red"), size = 2)
) + geom_point(show.legend = F) + theme_minimal() + ylab("") +
  ggtitle("Root Mean Squared Error is slightly lower when \n'Is Africa' is included as a variabl
e in the model")
```



The base models (intercept-only) clearly do poorly no matter what. But in every interation of model, the one that includes the Africa indicator has a lower RMSE than the one without the Africa indicator, even if the diffence in RMSE is slight. Excluding the is_africa indicator from the model results in a (roughly) 1% increase in any of the models' RMSE:

```
#calculate percentages of
(RMSE_h1_lm_more_noafrica - RMSE_h1_lm_nopartner)/RMSE_h1_lm_nopartner*100
```

```
## [1] 0.902915
```

```
(RMSE_h1_mm_int_noafrica - RMSE_h1_mm_int)/RMSE_h1_mm_int*100
```

```
## [1] 0.9030092
```

```
(RMSE_h1_lm_noafrica- RMSE_h1_lm_africa)/RMSE_h1_lm_noafrica*100
```

```
## [1] 0.9803678
```

3.1.4: Results

With a sample size of 200,000 observations in our test set and the consistency of this result across multiple related models, it seems clear that: 1) Being a loan applicant from Africa is predictive of becoming funded more slowly per $25 requested 2) The impact of this particular variable on overall funding speed is slight; I would not consider it practically significant, despite its statistical significance

In sum, the hypothesis is disproved but may not be a compelling factor in funding speed in the first place.

3.2: Hypothesis 2

Using a constructed sentiment score based on the “use” comments, we can predict (better than the null model) which loan requests will be fully-funded.

This hypothesis allows us to explore the predictive power of the text data that is part of the larger kiva dataset. Every funding request includes a “use” statement, which is why I have chosen that field over using the “tag” field. The overall percentage of projects that are not fully funded is small - around 7%, which means that both Kiva and requestors may be interested in factors correlated with success. As I have a personal interest in text data, that factor is one I have narrowed my focus to. I am curious if factors relating to how the projects are written about could affect their overall funding status.

```
#raw numbers
table(kiva_eda$fully_funded)
```

```
##
##     No    Yes
##   9817 124424
```

```
#percentage of the whole
table(kiva_eda$fully_funded)/nrow(kiva_eda)*100
```

```
##
##        No       Yes
##  7.312967 92.687033
```

```
#number of empty "use" fields
kiva_eda %>% filter(use == "") %>% nrow()
```

```
## [1] 844
```

```
#number of empty "tags" fields
kiva_eda %>% filter(tags == "") %>% nrow()
```

```
## [1] 34210
```

3.2.1: Exploratory Data Analysis

Let’s explore words and their relationship to funding status.
What do the “use” descriptions look like?

```
head(kiva_eda$use, 10)
```

```
##  [1] "to purchase 5 sacks of millet, 2 containers of oil, and 2 bags of sugar to make porridg
e to sell"
##  [2] "to buy additional materials like shells, nylon thread, wire, etc., for her business"

##  [3] "to pay university fees."

##  [4] "buy additional textiles, buttons, and sewing supplies"

##  [5] "to buy a water filter to provide safe drinking water for their family."

##  [6] "To buy a water filter to provide safe drinking water for their family.\t\t"

##  [7] "to buy a solar lantern."

##  [8] "to buy cattle."

##  [9] "to buy materials to sew more clothes to sell"

## [10] "to purchase material and supplies to process food for sale"
```

```
tail(kiva_eda$use, 10)
```

```
##  [1] "to buy shoes."

##  [2] ""

##  [3] "to purchase sand, cement and ballast for repairs"

##  [4] "to buy insecticides, fertilizer, pest control, and improved corn and sorghum seed to pl
ant during the next season"
##  [5] "to buy drinks to sell such as beer, wine, soft drinks and others."

##  [6] "to buy construction materials (cement, windows, doors, and stone) to finish building he
r house."
##  [7] "to build water sources at their home to improve access to sanitation for their family."

##  [8] "to buy adapters, presses, water treatments, lime and other replacements to keep her cor
n milling equipment in good condition."
##  [9] "to buy sugar, beans, oil, soap, drinks, condiments and other basic need products that h
er clients request."
## [10] "to buy feed and vitamins for her pigs."
```

We see lots of repeated words and phrases, with much more specific information than we get from other fields like Sector or Activity.

Let’s get the EDA data in shape for some more exploration

```
#split out the relevant fields
by_words <- kiva_eda %>% select(id, use, fully_funded)

#tidy for sentiment analysis
library(tidytext)
words_eda <- by_words %>% unnest_tokens(output = word, input = use)
```

Let's look at some simple statistics

```
length(unique(words_eda$word))
```

```
## [1] 13241
```

```
nrow(words_eda)
```

```
## [1] 1412367
```

13,241 distinct words in the EDA set, with 1,412,367 total words suggests a lot of repeated words.

What are some of the top and bottom words among the "Yes" group?

```
words_eda %>% filter(fully_funded == "Yes") %>% group_by(word) %>% tally() %>% arrange(desc(n))
 %>% head()
```

```
## # A tibble: 6 x 2
##   word          n
##   <chr>     <int>
## 1 to       173380
## 2 buy       72823
## 3 and       71000
## 4 for       45334
## 5 her       38130
## 6 purchase  29741
```

```
words_eda %>% filter(fully_funded == "Yes") %>% group_by(word) %>% tally() %>% arrange(n)  %>% h
ead()
```

```
## # A tibble: 6 x 2
##   word      n
##   <chr> <int>
## 1 0         1
## 2 0,39      1
## 3 0.5       1
## 4 0.7       1
## 5 0.971     1
## 6 000f      1
```

```
words_eda %>% filter(fully_funded == "No") %>% group_by(word) %>% tally() %>% arrange(desc(n))
 %>% head()
```

```
## # A tibble: 6 x 2
##   word         n
##   <chr>    <int>
## 1 to       13459
## 2 and       6326
## 3 buy       5483
## 4 for       3438
## 5 a         2243
## 6 purchase  2230
```

```
words_eda %>% filter(fully_funded == "No") %>% group_by(word) %>% tally() %>% arrange(n)  %>% he
ad()
```

```
## # A tibble: 6 x 2
##   word       n
##   <chr>  <int>
## 1 0.393      1
## 2 1,000      1
## 3 1.75       1
## 4 108        1
## 5 12,500     1
## 6 120        1
```

We see that the bottom words include varous numbers, only appearing once. The top words show strong overlap, including: “to”, “buy”, “and”, “for”, “purchase”. It’s curious that “her” is a top word, by frequency, for “Yes”, but that “a” holds a similar place among “No”. There might be a gender issue at play. Where does “her” show up for “No”?

```
words_eda %>% filter(fully_funded == "No", word == "her" | word == "a") %>% group_by(word) %>%
  tally() %>% arrange(desc(n))  %>% head()
```

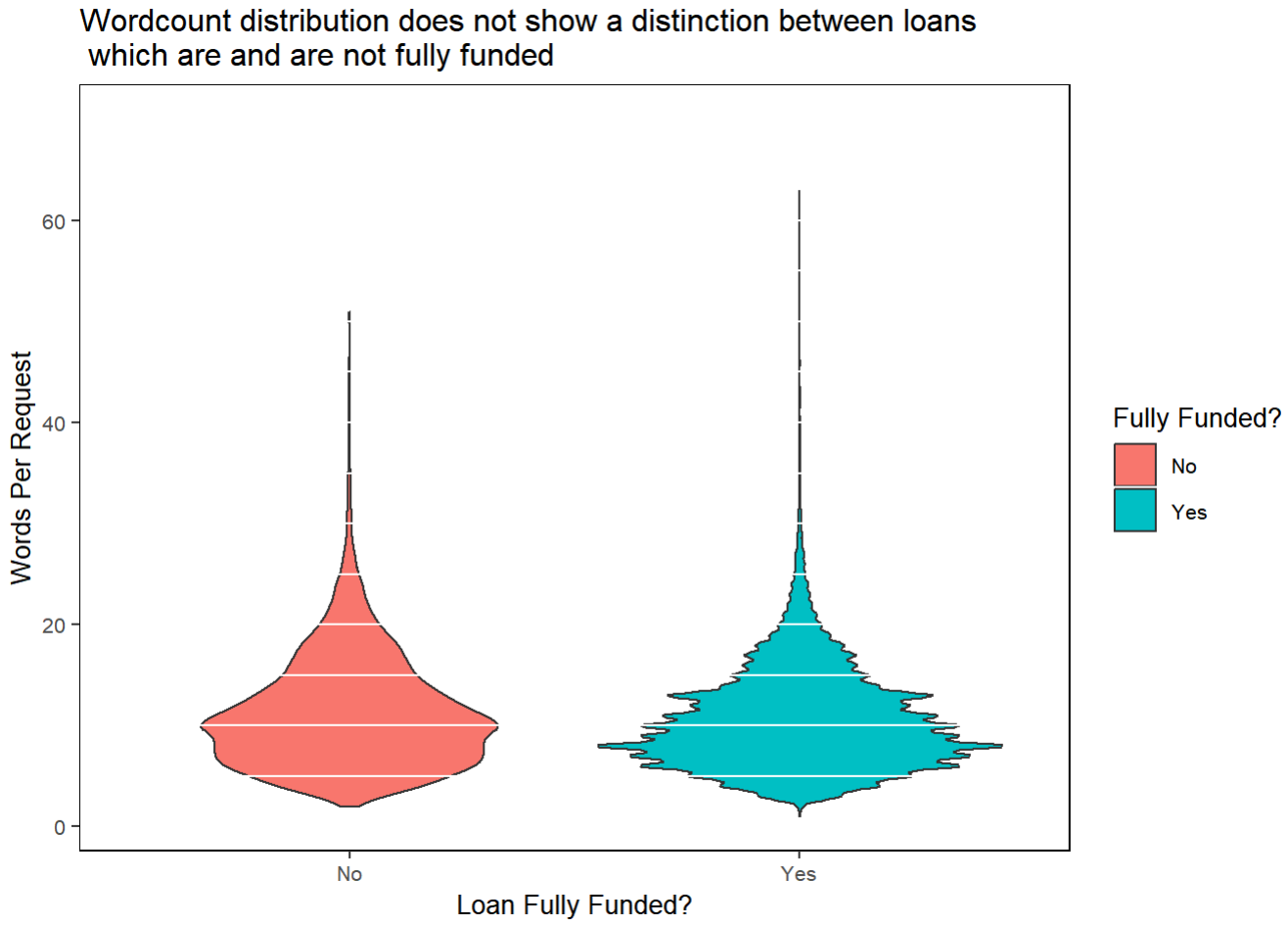
```
## # A tibble: 2 x 2
##   word      n
##   <chr> <int>
## 1 a      2243
## 2 her    2020
```

Both are frequently used, even though “a” has higher incidence.

The diffrence in n values between the “Yes” and “No” sets surely reflect the sheer number of fully-funded projects being much larger than the number of projects without full funding.

What is is distribution of word counts by funding status?

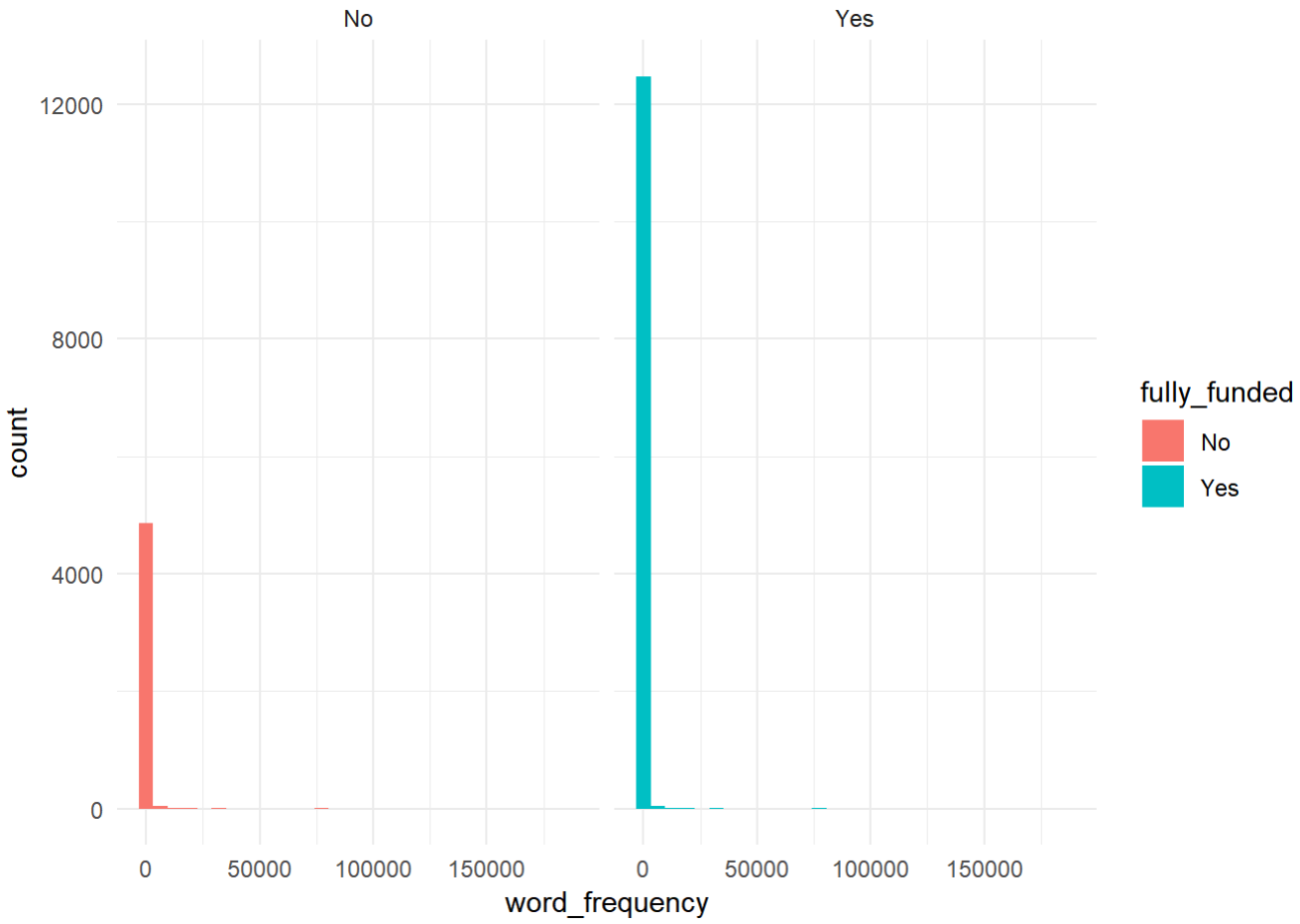
```
#calculate word count
words_eda <- words_eda %>% group_by(id) %>% mutate(word_count = n())

words_eda_graph <- words_eda %>% select(id, fully_funded, word_count) %>% distinct()

#graph it
ggplot(
  data = words_eda_graph,
  aes(x = fully_funded, y = word_count, fill = fully_funded)
) + geom_violin() +
  ylab("Words Per Request") + xlab("Loan Fully Funded?") +
  ggtitle("Wordcount distribution does not show a distinction between loans \n which are and are
 not fully funded") +
  theme(    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    text = element_text(size = 10)) +
  geom_hline(mapping=NULL, yintercept=seq(5,70, by = 5) ,colour='white') +
  scale_fill_discrete(name = "Fully Funded?")
```



What is the distribution of individual word frequency?

```
#calculate it
words_eda <- words_eda %>% group_by(word) %>% mutate(word_frequency = n())
words_eda_graph2 <- words_eda %>% select(word, word_frequency, fully_funded) %>% distinct()

#graph it
ggplot(
  words_eda_graph2,
  aes(x = word_frequency, fill = fully_funded)
) + geom_histogram() + theme_minimal() + facet_wrap(~fully_funded)
```

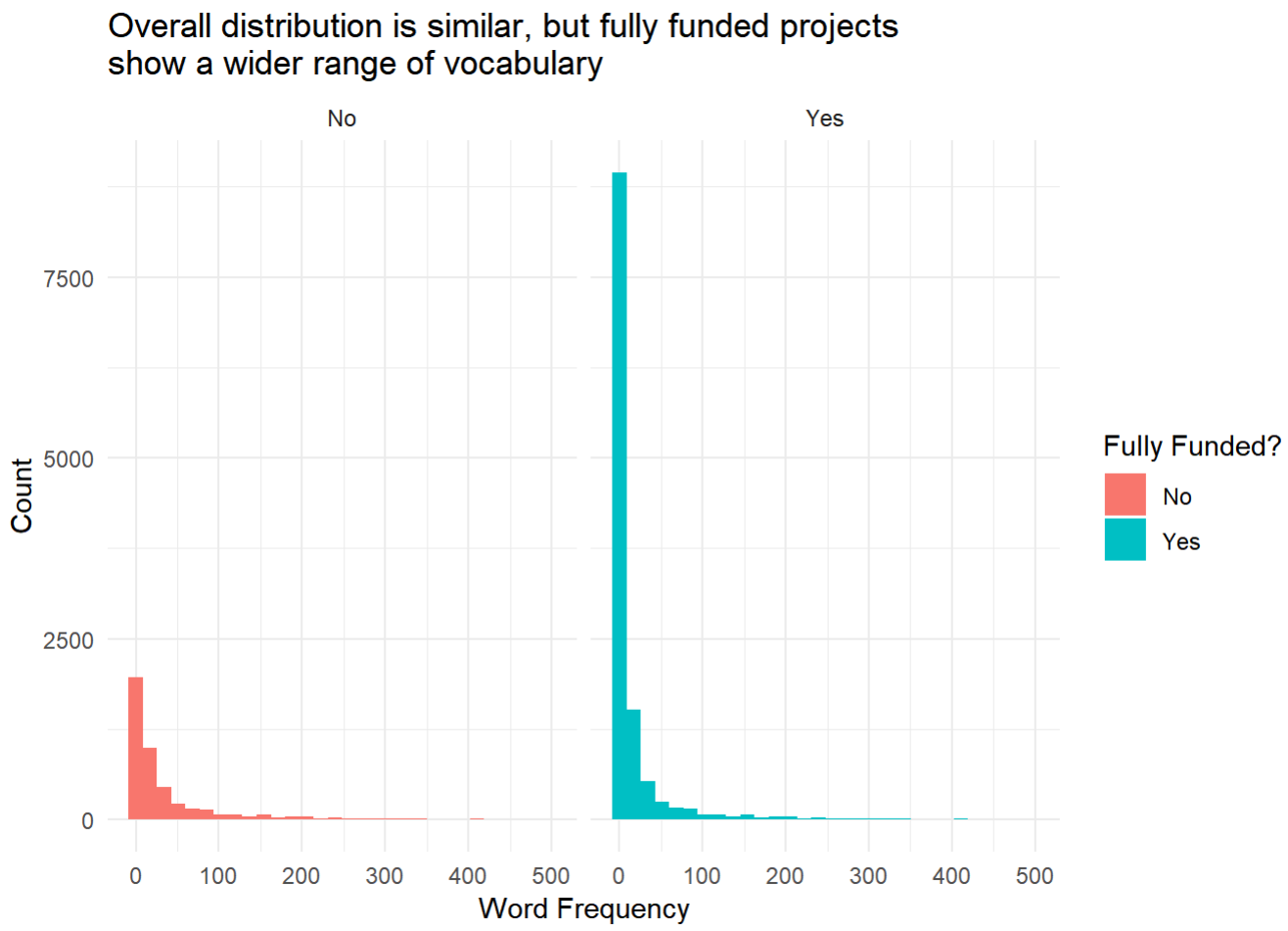
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Let's cut off the upper limit and see if we get a better sense of distribution

```
#graph it
ggplot(
  words_eda_graph2[words_eda_graph2$word_frequency < 500,],
  aes(x = word_frequency, fill = fully_funded)
) + geom_histogram() + theme_minimal() + facet_wrap(~fully_funded) +
  xlab("Word Frequency") + ylab("Count") + scale_fill_discrete("Fully Funded?") +
  ggtitle("Overall distribution is similar, but fully funded projects \nshow a wider range of vo
cabulary")
```

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



The fully funded requests show a higher rate of relatively infrequent words compared to the requests that do not obtain full funding.

3.2.2: Method

I will create a “sentiment” score using a subset of the training data, which will calculate a score for each word based on how often it is used in projects that receive full funding. I will try several calculations for these scores, including raw percentages of use and scores based on word frequency within words used for each funding outcome.

I will then apply these word scores to the other subset of the training and calculate average scores for the “use” sentences. After calculating the actual proportion of fully funded to not fully funded projects in the group, I will use that information to neatly divide the sample (ordered by score) into the correct number of “Yes” and “No” responses. This result (low scores = “No”, High scores = “Yes”) will be compared to the actual funding status of each sentence.

By using the various metrics that are part of the confusion matrix, I will evaluate the classification rate.

I will select whichever score calculation does best, rerun the setup on the entire training set, then apply the word scores to the test data. I will use the cutoff determined from teh training run to split the test data and see if the resulting classification can beat the null model.

3.2.3: Analysis

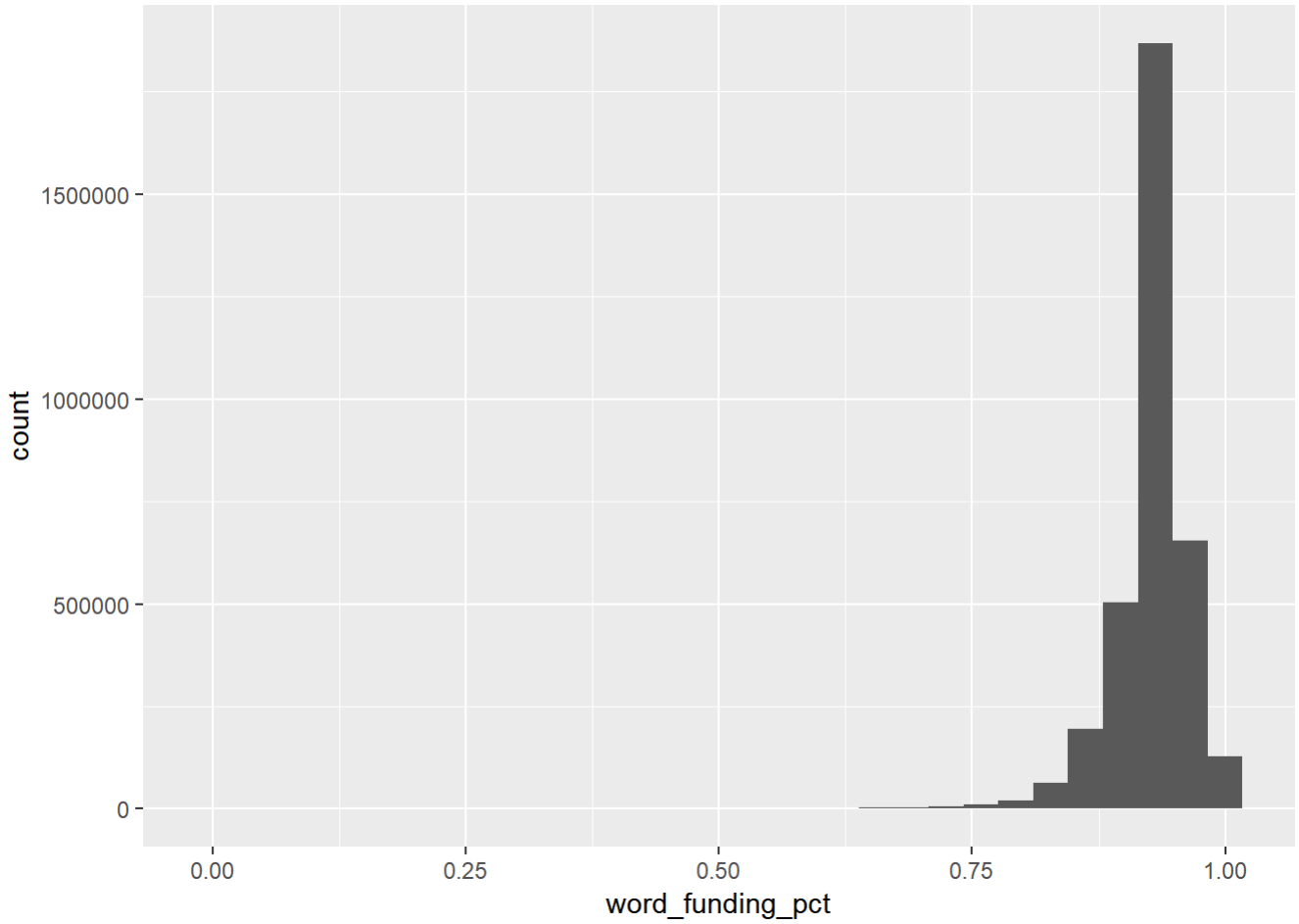
To prepare we set up the text train and test data; we will need to also split the train dataset into a traintrain and a traintest

```
#split the training data into a 70/30 split
set.seed(8553)
train_index <- sample(1:nrow(kiva_train), size = 0.7*nrow(kiva_train))
kiva_traintrain <- kiva_train[train_index,]
kiva_traintest <- kiva_train[-train_index,]

#set up words
#split out the relevant fields
by_words <- kiva_traintrain %>% select(id, use, fully_funded)
by_words2 <- kiva_traintest %>% select(id, use, fully_funded)

#tidy for sentiment analysis
words_train <- by_words %>% unnest_tokens(output = word, input = use)
words_test <- by_words2 %>% unnest_tokens(output = word, input = use)
```

Next, we use the test set to create funding-specific “sentiment”. We use all words to set up fractional sentiment

```
words_train <- words_train %>%
  mutate(full_fund_factor = ifelse(fully_funded == "Yes",1, 0)) %>% #make a funding factor
  group_by(word) %>%
  mutate(word_funding_pct = mean(full_fund_factor)) #new variable with average of its funding fa
ctor

#take a look
ggplot(
  dat = words_train,
  aes(x = word_funding_pct)
) + geom_histogram()
```

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




```
length(unique(words_train$word))
```

```
## [1] 19305
```

Now we use all words to set up a frequency-based sentiment, and will set a “more frequent” indicator to 1 or 0 depending on which situation is stronger. (For example, if “hello” has 0.001 frequence for “Yes” and 0.002 frequency for “No”, it will get an indicator of “0” for “No” and also a frequency score of -0.002 to indicate the higher value’s magnitude and direction)

```
#set up both new variables
words_train_frequency <- words_train %>% group_by(fully_funded) %>%
  mutate(full_fund_count = n()) %>% #overall count by funding status
  group_by(fully_funded, word) %>%
  mutate(word_count_by_funding = n()) %>% #overall count of each word within a funding status
  mutate(word_frequency = word_count_by_funding/full_fund_count) %>% #frequency of each word wit
hin a funding status
  group_by(word) %>%
  mutate(stronger_category = max(word_frequency)) %>% #pull out the bigger frequency for each wo
rd (Y vs N freq)
  select(fully_funded, word, stronger_category, word_frequency) %>%  #simplify the data
  filter(stronger_category == word_frequency) %>% #get rid of mismatches (e.g keep only the fund
ing status for each word that has a higher frequency)
  distinct() %>% #get rid of duplicates
  mutate(word_funding_freq_indicator = ifelse(fully_funded == "Yes", 1, 0)) %>% #convert to indi
cator
  mutate(word_freq_by_funding = ifelse(fully_funded == "Yes", word_frequency, -1*word_frequenc
y)) %>% #set frequency to a + and - scale (+ = Yes, - = No) to enable comparisons of magnitude l
ater
  select(word, word_freq_by_funding, word_funding_freq_indicator) #reduce to columns of interest
 and save


#add these new pieces to the full train dataset
words_train <- left_join(words_train, words_train_frequency)
```

```
## Joining, by = "word"
```

```
head(words_train)
```

```
## # A tibble: 6 x 7
## # Groups:   word [5]
##       id fully_funded word  full_fund_factor word_funding_pct
##    <int> <chr>        <chr>            <dbl>            <dbl>
## 1 1.01e6 Yes          to                   1            0.929
## 2 1.01e6 Yes          buy                  1            0.930
## 3 1.01e6 Yes          dess~                1            0.892
## 4 1.01e6 Yes          and                  1            0.918
## 5 1.01e6 Yes          swee~                1            0.847
## 6 6.68e5 No           to                   0            0.929
## # ... with 2 more variables: word_freq_by_funding <dbl>,
## #   word_funding_freq_indicator <dbl>
```

```
#remove dataset-specific fields and to one line per word
words_train_final <- words_train %>% select(-id, -fully_funded, - full_fund_factor) %>% distinct
()
```

Apply word “sentiment” scores of various types to the traintest data and see how the confusion matrices work

```
#setup
words_test <- left_join(words_test, words_train_final)
```

```
## Joining, by = "word"
```

```
head(words_test)
```

```
##        id fully_funded     word word_funding_pct word_freq_by_funding
## 1 1261858          Yes       to        0.9293099         0.1332329196
## 2 1261858          Yes purchase        0.9326237         0.0226887902
## 3 1261858          Yes        5        0.9390018         0.0001588772
## 4 1261858          Yes    sacks        0.9718234         0.0016935434
## 5 1261858          Yes       of        0.9256210         0.0200792009
## 6 1261858          Yes   millet        0.9006309        -0.0002386246
##   word_funding_freq_indicator
## 1                           1
## 2                           1
## 3                           1
## 4                           1
## 5                           1
## 6                           0
```

```
#base pct
words_base_pct <- words_test %>% group_by(id, fully_funded) %>% summarise(mean(word_funding_pct,
 na.rm = TRUE))
colnames(words_base_pct)[3] <- "sentence_by_word_pct"

#proportion of N to Y
cutoff1 <- (table(words_base_pct$fully_funded)/nrow(words_base_pct))[1]

#quantiles
base_pct_cutoff <- quantile(words_base_pct$sentence_by_word_pct, probs = cutoff1)
base_pct_cutoff
```

```
## 7.166821%
## 0.8965844
```

```
#predict based on true-value quantiles
words_base_pct$prediction <- ifelse(words_base_pct$sentence_by_word_pct < base_pct_cutoff, "No",
 "Yes")

#save as factors
words_base_pct$prediction <- as.factor(words_base_pct$prediction)
words_base_pct$fully_funded <- as.factor(words_base_pct$fully_funded)
```

```
#frequency
words_freq <- words_test %>% group_by(id, fully_funded) %>%
  summarise(mean(word_freq_by_funding, na.rm = TRUE))
colnames(words_freq)[3] <- "sentence_by_word_freq"

#proportion of N to Y
cutoff2 <- (table(words_freq$fully_funded)/nrow(words_freq))[1]

#quantiles
freq_cutoff <- quantile(words_freq$sentence_by_word_freq, probs = cutoff2)
freq_cutoff
```

```
##   7.166821%
## 0.008090718
```

```
#predict based on true-value quantiles
words_freq$prediction <- ifelse(words_freq$sentence_by_word_freq < freq_cutoff, "No", "Yes")

#save as factors
words_freq$prediction <- as.factor(words_freq$prediction)
words_freq$fully_funded <- as.factor(words_freq$fully_funded)
```

```
#frequency indicator
words_freq_indicator <- words_test %>% group_by(id, fully_funded) %>%
  summarise(mean(word_funding_freq_indicator, na.rm = TRUE))
colnames(words_freq_indicator)[3] <- "sentence_by_freq_indicator"

#proportion of N to Y
cutoff3 <- (table(words_freq_indicator$fully_funded)/nrow(words_freq_indicator))[1]

#quantiles
freq_indicator_cutoff <- quantile(words_freq_indicator$sentence_by_freq_indicator, probs = cutof
f3)
freq_indicator_cutoff
```

```
## 7.166821%
## 0.3333333
```

```
#Length(words_base_indicator$id[words_base_indicator$sentence_by_word_pct_indicator != 1])/nrow
(words_base_indicator)

#predict based on true-value quantiles
words_freq_indicator$prediction <- ifelse(words_freq_indicator$sentence_by_freq_indicator < freq
_indicator_cutoff, "No", "Yes")

#save as factors
words_freq_indicator$prediction <- as.factor(words_freq_indicator$prediction)
words_freq_indicator$fully_funded <- as.factor(words_freq_indicator$fully_funded)
```

Confusion matrices of results to measure classification

```
caret::confusionMatrix(data = words_base_pct$prediction, reference = words_base_pct$fully_funde
d, positive = "No")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     No    Yes
##        No    2148   7892
##        Yes   7892 122158
##
##                Accuracy : 0.8873
##                  95% CI : (0.8857, 0.889)
##     No Information Rate : 0.9283
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1533
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.21394
##             Specificity : 0.93932
##          Pos Pred Value : 0.21394
##          Neg Pred Value : 0.93932
##              Prevalence : 0.07167
##          Detection Rate : 0.01533
##    Detection Prevalence : 0.07167
##       Balanced Accuracy : 0.57663
##
##        'Positive' Class : No
##
```

```
caret::confusionMatrix(data = words_freq$prediction, reference = words_freq$fully_funded, positi
ve = "No")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     No    Yes
##        No    1216   8824
##        Yes   8824 121226
##
##                Accuracy : 0.874
##                  95% CI : (0.8723, 0.8758)
##     No Information Rate : 0.9283
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0533
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.12112
##             Specificity : 0.93215
##          Pos Pred Value : 0.12112
##          Neg Pred Value : 0.93215
##              Prevalence : 0.07167
##          Detection Rate : 0.00868
##    Detection Prevalence : 0.07167
##       Balanced Accuracy : 0.52663
##
##        'Positive' Class : No
##
```

```
caret::confusionMatrix(data = words_freq_indicator$prediction, reference = words_freq_indicator
$fully_funded, positive = "No")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     No    Yes
##        No    1691   7470
##        Yes   8349 122580
##
##                Accuracy : 0.8871
##                  95% CI : (0.8854, 0.8887)
##     No Information Rate : 0.9283
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1157
##  Mcnemar's Test P-Value : 0.000000000002935
##
##             Sensitivity : 0.16843
##             Specificity : 0.94256
##          Pos Pred Value : 0.18459
##          Neg Pred Value : 0.93623
##              Prevalence : 0.07167
##          Detection Rate : 0.01207
##    Detection Prevalence : 0.06539
##       Balanced Accuracy : 0.55549
##
##        'Positive' Class : No
##
```

These all do pretty terribly! The first word method (base percent) works the best, since it has the highest overall accuracy and the does the best job correctly predicting “No”, the more difficult class. Let’s include all of the training set data in the word ratings, then try the results on the complete test set.

```
#set up full training and test set words
#split out the relevant fields
kiva_train_words <- kiva_train %>% select(id, use, fully_funded)
kiva_test_words <- kiva_test %>% select(id, use, fully_funded)

#tidy for sentiment analysis
kiva_train_tidy_words <- kiva_train_words %>% unnest_tokens(output = word, input = use)
kiva_test_tidy_words <- kiva_test_words %>% unnest_tokens(output = word, input = use)
```

```
#rerun word scores with full training data
kiva_train_tidy_words <- kiva_train_tidy_words %>%
  mutate(full_fund_factor = ifelse(fully_funded == "Yes",1, 0)) %>% #make a funding factor
  group_by(word) %>%
  mutate(word_funding_pct = mean(full_fund_factor)) #new variable with average of its funding fa
ctor

kiva_train_tidy_words2 <- kiva_train_tidy_words %>%
  select(-id, -fully_funded, -full_fund_factor) %>% #reduce to variables of interest
  distinct() #get 1 row per word

head(kiva_train_tidy_words, 10)
```

```
## # A tibble: 10 x 5
## # Groups:   word [9]
##         id fully_funded word       full_fund_factor word_funding_pct
##      <int> <chr>        <chr>                 <dbl>            <dbl>
##  1 1261858 Yes          to                        1            0.929
##  2 1261858 Yes          purchase                  1            0.933
##  3 1261858 Yes          5                         1            0.943
##  4 1261858 Yes          sacks                     1            0.972
##  5 1261858 Yes          of                        1            0.926
##  6 1261858 Yes          millet                    1            0.888
##  7 1261858 Yes          2                         1            0.939
##  8 1261858 Yes          containers                1            0.949
##  9 1261858 Yes          of                        1            0.926
## 10 1261858 Yes          oil                       1            0.926
```

```
##############
#use training results to set quantiles

#base pct
train_base_pcts <- kiva_train_tidy_words %>% group_by(id, fully_funded) %>%
  summarise(mean(word_funding_pct, na.rm = TRUE))
colnames(train_base_pcts)[3] <- "sentence_by_word_pct"

#proportion of N to Y
cutoff <- (table(train_base_pcts$fully_funded)/nrow(train_base_pcts))[1]

#cutoff percentile
train_cutoff <- quantile(train_base_pcts$sentence_by_word_pct, probs = cutoff)
train_cutoff
```

```
## 7.164014%
## 0.8967245
```

```
##############
#join training scores with test set
kiva_test_tidy_words <- left_join(kiva_test_tidy_words, kiva_train_tidy_words2)
```

```
## Joining, by = "word"
```

```
#calculate sentence scores in test
test_base_pcts <- kiva_test_tidy_words %>% group_by(id, fully_funded) %>%
  summarise(mean(word_funding_pct, na.rm = TRUE))
colnames(test_base_pcts)[3] <- "sentence_by_word_pct"

#predict test based on training quantiles
test_base_pcts$prediction <- ifelse(test_base_pcts$sentence_by_word_pct < train_cutoff, "No", "Y
es")

#save as factors
test_base_pcts$prediction <- as.factor(test_base_pcts$prediction)
test_base_pcts$fully_funded <- as.factor(test_base_pcts$fully_funded)
```

```
caret::confusionMatrix(data = test_base_pcts$prediction, reference = test_base_pcts$fully_funde
d, positive = "No")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     No    Yes
##        No    3045  11265
##        Yes  11147 174600
##
##                Accuracy : 0.888
##                  95% CI : (0.8866, 0.8894)
##     No Information Rate : 0.9291
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.1534
##  Mcnemar's Test P-Value : 0.4345
##
##             Sensitivity : 0.21456
##             Specificity : 0.93939
##          Pos Pred Value : 0.21279
##          Neg Pred Value : 0.93999
##              Prevalence : 0.07094
##          Detection Rate : 0.01522
##    Detection Prevalence : 0.07153
##       Balanced Accuracy : 0.57697
##
##        'Positive' Class : No
##
```

Using the results from the training set to predict the test values, we see a similarly bad result for overall accuracy: 88.91%, worse than the null model at 92.88%. However, this model does identify 21.857% of requests that are not fully funded. This means that, compared to the null model, the calculated word scores do provide an improvement in prediction.

3.2.4: Results

The decisions to use text sentiment as a predictor depends on your goals. If overall accuracy is desired, the null model (or not bothering with sentiment) will have the best result. But the fact that we are able to find 21% of the actually-not-fully-funded requests suggests that there may be some predictive power buried within these scores. I would say that the consistency of results between the training and test data shows that some words or phrases really are associated more with full-funding vs. non-full funding. Let's take a look at some examples

```
#pull out lowest scores for non-funded projects
low_score_ids <- test_base_pcts %>% filter(fully_funded == "No") %>% arrange(sentence_by_word_pc
t) %>% head() %>% select(id)

#Look at their statements
kiva_test %>% filter(id %in% low_score_ids$id) %>% select(fully_funded, use)
```

```
##   fully_funded
## 1           No
## 2           No
## 3           No
## 4           No
## 5           No
## 6           No
##                                                                  use
## 1                           Pintarrajear su cuarto y comprarse una cama
## 2         Pretend the issue with loan got addressed by Kiva Coordinator.
## 3 Pretend the issue with spanish loan was addressed by Kiva Coordinator.
## 4                                          purchase a concession trailer
## 5 Pretend the issue with spanish loan was addressed by Kiva Coordinator.
## 6                         Pretend the flagged issue was addressed by KC.
```

These statements are fascinating – it looks like one of them is a true use for the loan, and the others are being used as notes fields in a way that might indicate something strange or underhanded. I can see how a word like "pretend" might have a low calculated score based on these responses. Let's check that too

```
kiva_train_tidy_words %>% filter(word == "pretend") %>% group_by(fully_funded, word_funding_pct)
 %>% tally()
```

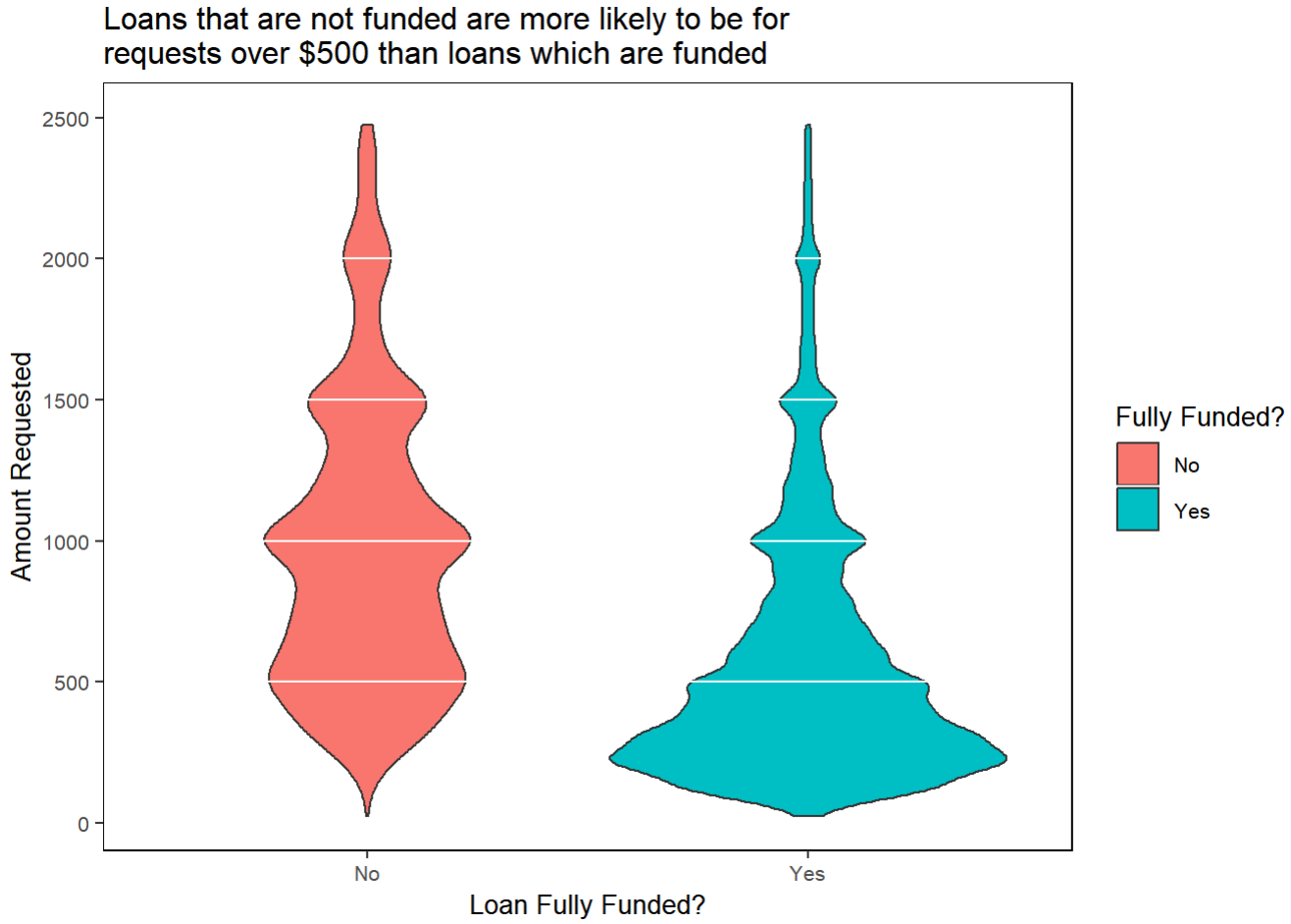
```
## # A tibble: 2 x 3
## # Groups:   fully_funded [?]
##   fully_funded word_funding_pct     n
##   <chr>                   <dbl> <int>
## 1 No                      0.111     8
## 2 Yes                     0.111     1
```

The training set here seems to have correctly classified at least one word as clearly associated with requests that are not fully funded.

Appendix 4: Extra Bits

Extras analysis from the random-forest-prep work for Hypothesis 2 (avenue only somewhat pursued; some code included but not run)

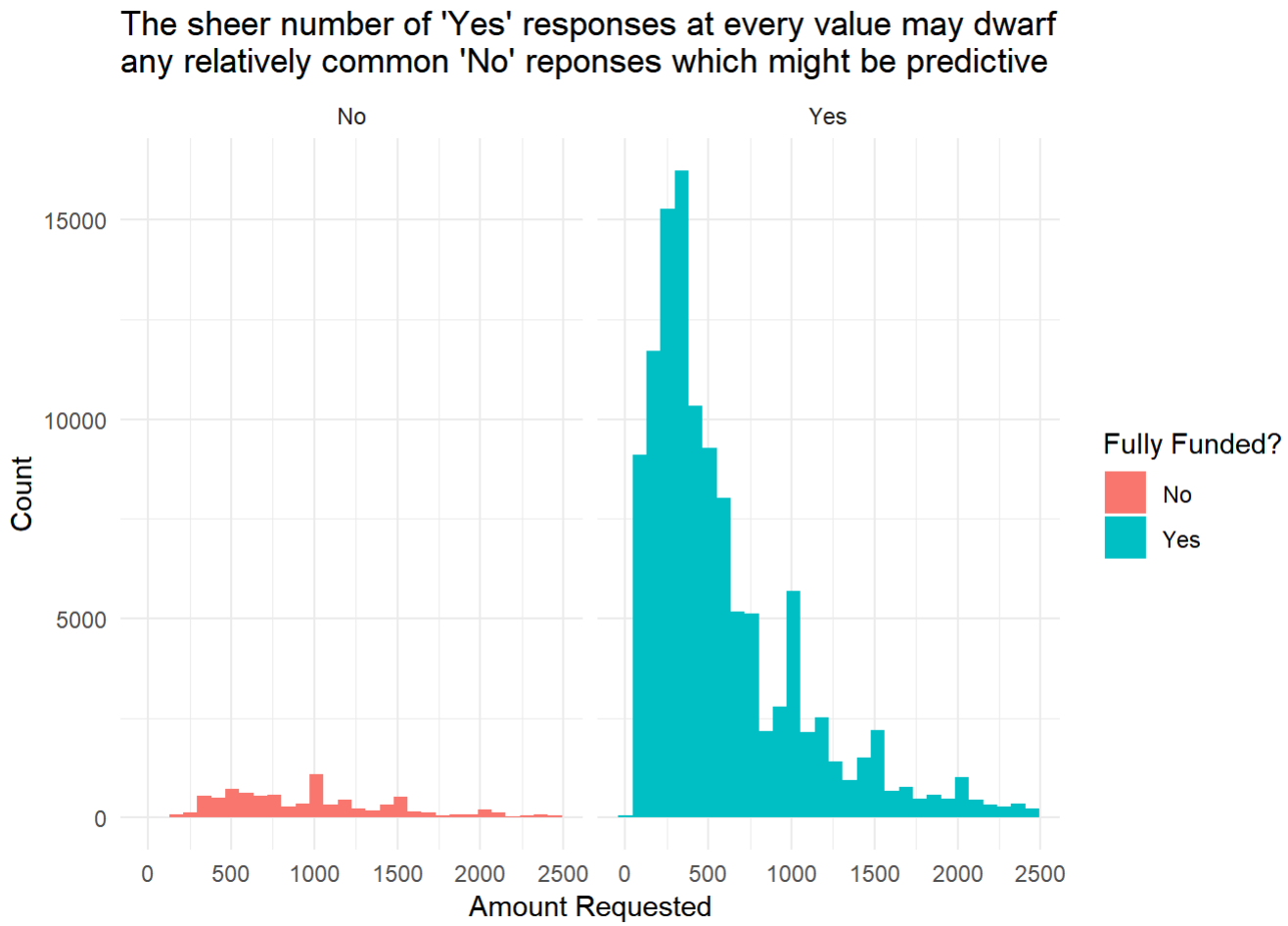
```
ggplot(
  data = kiva_eda[kiva_eda$loan_amount  < 2500, ],
  aes(x = fully_funded, y = loan_amount, fill = fully_funded)
) + geom_violin() +
  ylab("Amount Requested") + xlab("Loan Fully Funded?") +
  ggtitle("Loans that are not funded are more likely to be for \nrequests over $500 than loans w
hich are funded") +
  theme(    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    text = element_text(size = 10)) +
  geom_hline(mapping=NULL, yintercept=seq(500,2500, by = 500) ,colour='white') +
  scale_fill_discrete(name = "Fully Funded?")
```



Regardless of funding status, the two shapes show clearly that requests under $500 are more likely to be fully-funded than requests over $500. In fact, at the low end, it seems like nearly all requests are fully funded. At the upper end, the difference seems less extreme. This image, however, is based on percentages within each group. What if we look at the actual number of requests at each point dollar amount?

```
ggplot(
  data = kiva_eda[kiva_eda$loan_amount < 2500],
  aes(x = loan_amount, fill = fully_funded)
) + geom_histogram() + facet_wrap(~fully_funded) +
  ylab("Count") + xlab("Amount Requested") + ggtitle("The sheer number of 'Yes' responses at eve
ry value may dwarf \nany relatively common 'No' reponses which might be predictive") +
  scale_fill_discrete(name = "Fully Funded?") +
  theme_minimal()
```

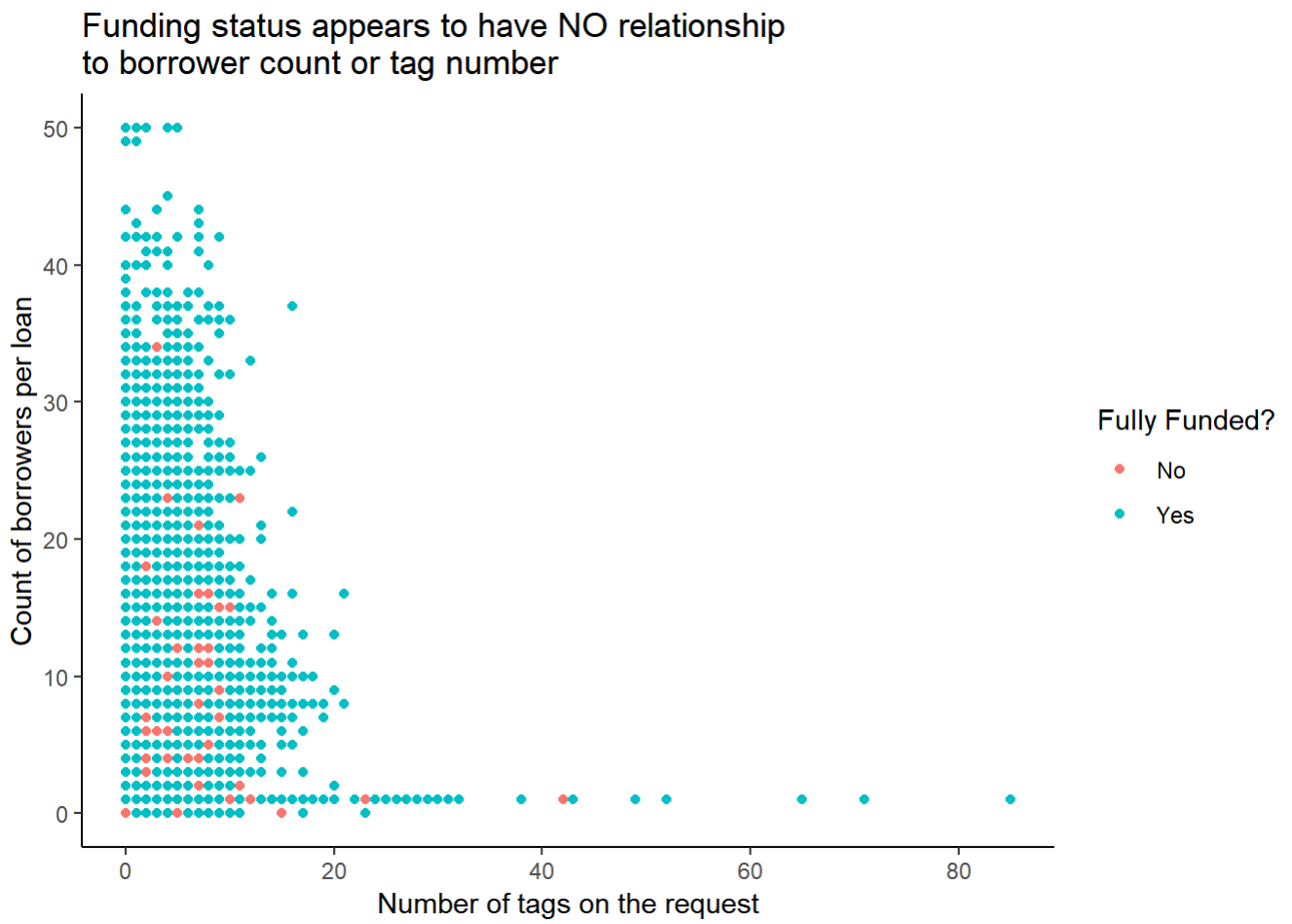
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Because so few loan requests are for values over $1000, it may mean that we are only able to predict effectively at the upper tails. So many of the low-value loans are fully-funded that they ought to be virtually impossible to distinguish from those that don't receive all their requested funds.

Let's try another variable, for luck:

```
ggplot(
  data = kiva_eda,
  aes(x = number_of_tags, y = borrowerCount_total, color = fully_funded)
) + geom_point() +
  ylab("Count of borrowers per loan") + xlab("Number of tags on the request") +
  ggtitle("Funding status appears to have NO relationship \nto borrower count or tag number") +
  scale_color_discrete(name = "Fully Funded?") +
  theme_classic()
```



The scattering of red dots (not fully funded) doesn't appear to follow any particular pattern. The combination of these variables (number of borrowers per loan and number of tags) does not appear to be related to funding status. Therefore, they are unlikely to be useful predictors.

With these starting images, it seems like there's a chance that the request amount might help with prediction at the upper end of dollar amounts, but that other variables may not have predictive power when determining if a project is fully-funded. Let's see what the models show.

Funding by sector?

```
kiva_eda %>% group_by(fully_funded, sector) %>% tally() %>% arrange(sector)
```

```
## # A tibble: 30 x 3
## # Groups:   fully_funded [2]
##    fully_funded sector           n
##    <chr>        <chr>        <int>
##  1 No           Agriculture   2832
##  2 Yes          Agriculture  33270
##  3 No           Arts            48
##  4 Yes          Arts          2376
##  5 No           Clothing       636
##  6 Yes          Clothing      5908
##  7 No           Construction    72
##  8 Yes          Construction  1251
##  9 No           Education      156
## 10 Yes          Education     6036
## # ... with 20 more rows
```

What about as a percentage of the sector?

```
kiva_eda %>% group_by(sector) %>% mutate(sector_count = n()) %>%
  ungroup() %>%
  group_by(fully_funded, sector, sector_count) %>% tally() %>%
  mutate(sector_pct = round(n/sector_count*100)) %>% select(fully_funded, sector, sector_pct) %
>%
  arrange(sector)
```

```
## # A tibble: 30 x 3
## # Groups:   fully_funded, sector [30]
##    fully_funded sector       sector_pct
##    <chr>        <chr>             <dbl>
##  1 No           Agriculture           8
##  2 Yes          Agriculture          92
##  3 No           Arts                  2
##  4 Yes          Arts                 98
##  5 No           Clothing             10
##  6 Yes          Clothing             90
##  7 No           Construction          5
##  8 Yes          Construction         95
##  9 No           Education             3
## 10 Yes          Education            97
## # ... with 20 more rows
```

All sectors show a high percentage of funded projects, but with a wider range than I had anticipated. The percentage of full-funding ranges from 87%-99% depending on the sector, with manufacturing at the high end and entertainment at the low end.

Analysis of related questions about full funding begins (code not run)

```
#set fully_funded as a factor
kiva_train$fully_funded <- ifelse(kiva_train$fully_funded == "Yes",1,0)
```

Linear model

As with hypothesis 1, we'll start with a base model. To avoid correlated variables, I'm excluding funded_amount and looking only at loan_amount. The other variables are the same as in hypothesis 1, although I've also removed is_africa. Later, we'll add in a full location variable; however, because these variables are large factors I'd like to run a simpler model first.

```
mod_h2_lm_base <- lm(fully_funded ~
                       loan_amount +
                       term_in_months +
                       lender_count +
                       repayment_interval +
                       borrowerCount_female +
                       borrowerCount_male +
                       number_of_tags,
                     dat = kiva_train)


summary(mod_h2_lm_base)
```

Even though all of these variables show strong statistical significance, their beta-hats are tiny! They look like they have very little influence on the model, with the repayment_interval factor possibly making the most difference.

Let's try another model that only uses that value as a predictor

```
mod_h2_lm_repint <- lm(fully_funded ~ repayment_interval, data = kiva_train)

summary(mod_h2_lm_repint)
```

As the lone variable, this factor does have a slightly larger influence on fully_funded, but is dwarfed by the intercept.

We'll try one more straightforward linear model, based on the partner involved with the requestor. I'd like to only have one variable (in a linear model) that indicates location, and as we saw above, partner_id appears to meet that criteria. As a contextual note, it's plausible that some partners have higher success rates than others when it comes to funding, especially because some partners will pre-fund a request before it is officially funded by the public.

```
#first, factorize parter_id
kiva_train$partner_id <- as.factor(kiva_train$partner_id)

#then run the model
mod_h2_lm_partner <- lm(fully_funded ~ partner_id, data = kiva_train)

#this output is going to be ridiculous!
summary(mod_h2_lm_partner)
```

It's not surprising that this model doesn't fare particularly well. Although some of the partners seems to have a statistically-significant effect on the prediction, they have very slight practical effects on the model. There may be too many partners to effectively work with in OLS.

None of the OLS models so far have done a particularly good job at producing a useful-looking model. The first model appears to have the most promise, with the highest Adjusted R^2 (0.15), but this level of variation explained may be insufficient to correctly predict a category comprising only 8%, more or less, of the populuation.

Mixed model Because the repayment_interval did show some promise, let’s see how it does for us within a mixed model. We’ll use it a a random intercept with the original group of linear model predictors.

```
library(lme4)
mod_h2_mm_int <- lmer(fully_funded ~
                        loan_amount +
                        term_in_months +
                        lender_count +
                        borrowerCount_female +
                        borrowerCount_male +
                        number_of_tags +
                         (1|repayment_interval), data = kiva_train)

summary(mod_h2_mm_int)
```

The random intercept only explains

```
cat(0.001472/(0.001472 + 0.054881)*100, "% of the variance due to random effects.", sep = "")
```

```
## 2.612106% of the variance due to random effects.
```

This suggests that, much like our linear model, a mixed model using similar variables will not be a productive predictor for determining if a request is going to be fully_funded.

With this table of models, let’s make predictions and calculate accuracy metrics.

```
pred_h2_lm_base <- predict(object = mod_h2_lm_base,
                           newdata = kiva_test, type = "response")
#round to 0 and 1
pred_h2_lm_base <- ifelse(pred_h2_lm_base >= 0.5, 1, 0)

pred_h2_lm_repint <- predict(object = mod_h2_lm_repint,
                           newdata = kiva_test, type = "response")
#round to 0 and 1
pred_h2_lm_repint <- ifelse(pred_h2_lm_repint >= 0.5, 1, 0)

#####THIS ONE ISN"T WORKING YET
#kiva_test$partner_id <- as.factor(kiva_test$partner_id)
#pred_h2_lm_partner <- predict(object = mod_h2_lm_partner,
#                            newdata = kiva_test, type = "response")

#unique(kiva_train$partner_id)
#unique(kiva_test$partner_id)

#round to 0 and 1
#pred_h2_lm_partner <- ifelse(pred_h2_lm_partner >= 0.5, 1, 0)

#this one is fine
pred_h2_mm_int <- predict(object = mod_h2_mm_int,
                           newdata = kiva_test, type = "response")
#round to 0 and 1
pred_h2_mm_int <- ifelse(pred_h2_mm_int >= 0.5, 1, 0)

cbind(pred_h2_lm_base, pred_h2_lm_repint, pred_h2_mm_int) %>%  summary()
```

```
#check various results as they come out, for weird stuff
table(pred_h2_lm_base)
table(kiva_test$fully_funded)
```

Calculate metrics

```
library(caret)


pred_h2_lm_base <- as.factor(ifelse(pred_h2_lm_base == 1, "Yes", "No"))
cm_h2_lm_base <- confusionMatrix(data = pred_h2_lm_base, reference = kiva_test$fully_funded, pos
itive = "Yes")

pred_h2_lm_repint <- as.factor(ifelse(pred_h2_lm_repint == 1, "Yes", "No"))
cm_h2_lm_repint <- confusionMatrix(data = pred_h2_lm_repint, reference = kiva_test$fully_funded,
 positive = "Yes")

#pred_h2_lm_partner <- as.factor(ifelse(pred_h2_lm_partner == 1, "Yes", "No"))
#cm_h2_lm_partner <- confusionMatrix(data = pred_h2_lm_partner, reference = kiva_test$fully_fund
ed, positive = "Yes")

pred_h2_mm_int <- as.factor(ifelse(pred_h2_mm_int == 1, "Yes", "No"))
cm_h2_mm_int <- confusionMatrix(data = pred_h2_mm_int, reference = kiva_test$fully_funded, posit
ive = "Yes")

accuracy_metrics <- as.data.frame(matrix(0, ncol=4, nrow=4))
colnames(accuracy_metrics) <- c("ModelType", "Accuracy", "Specificity", "Sensitivity")

accuracy_metrics[,1] <- c("BaseModel", "RepaymentInterval", "PartnerID", "MixedModel")

accuracy_metrics[1,2] <- cm_h2_lm_base$overall["Accuracy"]
accuracy_metrics[1,3] <- cm_h2_lm_base$byClass["Specificity"]
accuracy_metrics[1,4] <- cm_h2_lm_base$byClass["Sensitivity"]

accuracy_metrics[2,2] <- cm_h2_lm_repint$overall["Accuracy"]
accuracy_metrics[2,3] <- cm_h2_lm_repint$byClass["Specificity"]
accuracy_metrics[2,4] <- cm_h2_lm_repint$byClass["Sensitivity"]

accuracy_metrics[3,2] <- cm_h2_lm_partner$overall["Accuracy"]
accuracy_metrics[3,3] <- cm_h2_lm_partner$byClass["Specificity"]
accuracy_metrics[3,4] <- cm_h2_lm_partner$byClass["Sensitivity"]

accuracy_metrics[4,2] <- cm_h2_mm_int$overall["Accuracy"]
accuracy_metrics[4,3] <- cm_h2_mm_int$byClass["Specificity"]
accuracy_metrics[4,4] <- cm_h2_mm_int$byClass["Sensitivity"]

accuracy_metrics

table(kiva_test$fully_funded)/nrow(kiva_test)
```

Let’s move on to other models that will allow us to work with a larger group of data.

Random forest

A random forest classification is appealing because it will do a better job handling our fields with extensive factors. We’ll run several repetitions of different kinds of random forests together in a single loop. This method ensures that for each iteration, every type of random forest is using the same batch of data so the comparisons bewteen setups are fair.

The different kinds of forests we'll run are: 1) Using the same set of predictors as each linear model 2) Using all predictors that are numeric or factor-based 3) Using a subset of numeric/factor predictors that represent less overlapping information

After comparing errors, we'll re-run the best option with additional tuning of parameters like mtry and weights. First, we need to prepare the data into a format that will work with randomForest. This means turning character vectors into categoricals. It also means doing a dimensional reduction on factors with a lot of levels. As it turns out, the randomForest function can'thandle categorical predictors with more than 53 categories (thanks, error message!).

Update data types

```
#reset fully_funded to a character factor
#kiva_train$fully_funded <- ifelse(kiva_train$fully_funded == 1, "Yes", "No")
#kiva_train$fully_funded <- as.factor(kiva_train$fully_funded)

#turn other characters into factors
kiva_train$activity <- as.factor(kiva_train$activity)
kiva_train$sector <- as.factor(kiva_train$sector)
kiva_train$country_code <- as.factor(kiva_train$country_code)
kiva_train$country <- as.factor(kiva_train$country)
kiva_train$region <- as.factor(kiva_train$region)
kiva_train$currency <- as.factor(kiva_train$currency)
kiva_train$repayment_interval <- as.factor(kiva_train$repayment_interval)
```

Dimensional reduction
Factors with issues are:
-activity (163 levels)
-country_code (82 levels)
-country (82 levels)
-region (11321 levels)
-currency (67 levels)
-partner_id (354 levels)

We know that some of these are nested subsets. Activity is a subset of Sector (which only has 15 categories). Region is a subset of country/country_code (with 82 categories). Partner_id also looks like it might be a subset of country, based on our previous analysis of that factor. Currency is related to country, but is larger. So we might say we have:

currency > country/country_code > partner_id > region
sector > activity

For now, we'll use sector (ok already) and dummy variables for country in the models, excluding others.

```
#Let's try to turn country into a series of dummy variables
kiva_train_dummies <- kiva_train %>%
  unnest(country) %>%
  mutate(new = 1) %>%
  spread(country, new, fill = 0)

#now: reduce the df to only the variables we'll end up using (eventually) in the RF analysis
kiva_train_dummies <- kiva_train_dummies %>% select(-c(id, funded_amount, activity,
                                                       use, country_code, region, currency,
                                                       partner_id, posted_time,
                                                       disbursed_time, funded_time,
                                                       tags, borrower_genders, date,
                                                       is_africa))

#to get the RFs to work properly, we need to make sure that every included country has some numb
er of fully_funded and not fully_funded results

kiva_train %>% group_by(country, fully_funded) %>% tally() %>% arrange(country, fully_funded)

#let's set up a factor that combines information about country and funding.
kiva_train_dummies$country_funded <- as.factor(paste0(kiva_train$country, kiva_train$fully_funde
d))

#get the names right
colnames(kiva_train_dummies) <- gsub(" ", "_", colnames(kiva_train_dummies))
colnames(kiva_train_dummies) <- gsub("'", "_", colnames(kiva_train_dummies))
colnames(kiva_train_dummies) <- gsub(")", "_", colnames(kiva_train_dummies))
colnames(kiva_train_dummies) <- gsub("\\(", "_", colnames(kiva_train_dummies))
colnames(kiva_train_dummies) <- gsub("-", "_", colnames(kiva_train_dummies))

kiva_train_dummies %>% group_by(country_funded) %>% tally() %>% arrange(n)
```

```
#set replications
R <- 50

#set mtry tuning
#grid_mtry <- expand.grid(mtry=c(1:10))
library(caret)
library(rminer)
library(randomForest)

#set up control method
fitControl <- trainControl(method = "cv",  #do we care about CV?  I guess it doesn't hurt
                           number = 5,
                           returnData = TRUE,
                           returnResamp = "final",
                           summaryFunction = twoClassSummary,
                           classProbs = TRUE)

# create the error matrices to store values
err_mat_accuracy <- as.data.frame(matrix(0, ncol=4, nrow=R))
err_mat_sens <- as.data.frame(matrix(0, ncol=4, nrow=R))
err_mat_spec <- as.data.frame(matrix(0, ncol=4, nrow=R))

#create a list to store importance
imp_rf_base <- vector("list", R)
imp_rf_repint <- vector("list", R)
imp_rf_country <- vector("list", R)
imp_rf_all <- vector("list", R)

#############
### LOOP
#############
for(i in 1:R) {
  #create training and test set, 60/40 ratio, stratified samples
  id <- holdout(kiva_train_dummies$country_funded,
                ratio=.6,
                mode='stratified')
  select_train <- id$tr
  select_test <- id$ts
  kiva_tr <- kiva_train_dummies[select_train,]
  kiva_te <- kiva_train_dummies[select_test,]


  ##########
  # RANDOM FOREST: base LM
  ##########

  #make it have 100 trees

  rf_base = randomForest(fully_funded ~ loan_amount +
                         term_in_months +
                         lender_count +
                         repayment_interval +
                         borrowerCount_female +
```

```
                     borrowerCount_male +
                     number_of_tags,
                   kiva_tr,
                   ntree = 100,
                   trControl = fitControl,
                   metric = "ROC") #,
                   #tuneGrid = grid_mtry)

#reduce test data to variables of interest
kiva_te_base <- kiva_te %>% select(loan_amount, term_in_months, lender_count,
                                   repayment_interval, borrowerCount_female,
                                   borrowerCount_male,
                                   number_of_tags)

#predict
pred_rf_base <- predict(rf_base, kiva_te_base)



#calculate errors and save to matrices

#calc confusion
cm_rf_base <- confusionMatrix(pred_rf_base,
                         kiva_te$fully_funded, positive = "Yes")

#sensitivity
err_mat_sens[i,1] <- cm_rf_base$byClass["Sensitivity"]

#specificity
err_mat_spec[i,1] <- cm_rf_base$byClass["Specificity"]


#turn factors back into characters for comparison purpose
pred_rf_base <- as.character(pred_rf_base)
kiva_te_base$fully_funded <- as.character(kiva_te$fully_funded)

#accuracy
err_mat_accuracy[i,1] <- 1 - mean(pred_rf_base != kiva_te_base$fully_funded)

#save importance
imp_rf_base[[i]] <- importance(rf_base)

#########
# RANDOM FOREST: repayment interval LM
#########

#make it have 100 trees

rf_repint = randomForest(fully_funded ~
                     repayment_interval,
                   kiva_tr,
                   ntree = 100,
                   trControl = fitControl,
                   metric = "ROC")#,
```

```
                      #tuneGrid = grid_mtry)

  #reduce test data to variables of interest
  kiva_te_repint <- kiva_te %>% select(repayment_interval)

  #predict
  pred_rf_repint <- predict(rf_repint, kiva_te_repint)



  #calculate errors and save to matrices


  #calc confusion
  cm_rf_repint <- confusionMatrix(pred_rf_repint,
                           kiva_te$fully_funded, positive = "Yes")

  #sensitivity
  err_mat_sens[i,2] <- cm_rf_repint$byClass["Sensitivity"]

  #specificity
  err_mat_spec[i,2] <- cm_rf_repint$byClass["Specificity"]

  #turn factors back into characters for comparison purpose
  pred_rf_repint <- as.character(pred_rf_repint)
  kiva_te_repint$fully_funded <- as.character(kiva_te$fully_funded)

  #accuracy
  err_mat_accuracy[i,2] <- 1 - mean(pred_rf_repint != kiva_te_repint$fully_funded)

  #save importance
  imp_rf_repint[[i]] <- importance(rf_base_repint)

  #########
  # RANDOM FOREST: using country, instead of parter_id
  #########

  #make it have 100 trees

  #reduce training data to variables of interest
  kiva_tr_country <- kiva_tr %>% select(-c(loan_amount, sector, term_in_months, lender_count,
                                           repayment_interval, borrowerCount_female, borrowerCou
nt_male,
                                           borrowerCount_total, time_to_fund, time_to_fund_per_2
5,
                                           number_of_tags, country_funded))


  rf_country = randomForest(fully_funded ~ .,
                       kiva_tr_country,
                       ntree = 100)#,
                       #trControl = fitControl, #removing cross-validation for run-time
                       #metric = "ROC")#,
                       #tuneGrid = grid_mtry)
```

```
  #reduce testing data to variables of interest
  kiva_te_country <- kiva_te %>% select(-c(loan_amount, sector, term_in_months, lender_count,
                                           repayment_interval, borrowerCount_female, borrowerCou
nt_male,
                                           borrowerCount_total, time_to_fund, time_to_fund_per_2
5,
                                           number_of_tags, fully_funded))

  #predict
  pred_rf_country <- predict(rf_country, kiva_te_country)


  #calculate errors and save to matrices

  #calc confusion
  cm_rf_country <- confusionMatrix(pred_rf_country,
                           kiva_te$fully_funded, positive = "Yes")

  #sensitivity
  err_mat_sens[i,3] <- cm_rf_country$byClass["Sensitivity"]

  #specificity
  err_mat_spec[i,3] <- cm_rf_country$byClass["Specificity"]

  #turn factors back into characters for comparison purpose
  pred_rf_country <- as.character(pred_rf_country)
  kiva_te_country$fully_funded <- as.character(kiva_te$fully_funded)

  #accuracy
  err_mat_accuracy[i,3] <- 1 - mean(pred_rf_country != kiva_te_country$fully_funded)

  #save importance
  imp_rf_country[[i]] <- importance(rf_base_country)

  ##########
  # RANDOM FOREST: using "all" variables
  ##########

  #make it have 100 trees

  #reduce training data to variables of interest
  kiva_tr_all <- kiva_tr %>% select(-c(country_funded, time_to_fund, time_to_fund_per_25))


  rf_all = randomForest(fully_funded ~ .,
                        kiva_tr_all,
                        ntree = 100)#,
                        #trControl = fitControl, #removing cross-validation for run-time
                        #metric = "ROC")#,
                        #tuneGrid = grid_mtry)
```

```
#reduce testing data to variables of interest
kiva_te_all <- kiva_te %>% select(-c(fully_funded, country_funded))

#predict
pred_rf_all <- predict(rf_all, kiva_te_all)


#calculate errors and save to matrices

#calc confusion
cm_rf_all <- confusionMatrix(pred_rf_all,
                         kiva_te$fully_funded, positive = "Yes")

#sensitivity
err_mat_sens[i,4] <- cm_rf_all$byClass["Sensitivity"]

#specificity
err_mat_spec[i,4] <- cm_rf_all$byClass["Specificity"]

#turn factors back into characters for comparison purpose
pred_rf_all <- as.character(pred_rf_all)
kiva_te_all$fully_funded <- as.character(kiva_te$fully_funded)

#accuracy
err_mat_accuracy[i,4] <- 1 - mean(pred_rf_all != kiva_te_all$fully_funded)

#save importance
imp_rf_all[i]] <- importance(rf_all)

#
# #########
# # WEIGHTED RANDOM FOREST -- not prepped yet
# #########
#
# #also have 100 trees
# rfw = randomForest(Class ~ .,
#                         kiva_tr,
#                         ntree = 100,
#                         trControl = fitControl,
#                         metric = "ROC",
#                         classwt=c(.05,.95),
#                         tuneGrid = grid_mtry)
# #predict
# pred_rfw <- predict(rfw, kiva_te[-35])
#
# #calculate errors and save to matrices
#
# #accuracy
# err_mat_accuracy[i,4] <- 1 - mean(pred_rfw != kiva_te[,35])
#
# #calc confusion
# cm_rfw <- confusionMatrix(pred_rfw, kiva_te[[35]],
#                             positive = "good")
#
```

```
  # #sensitivity
  # err_mat_sens[i,4] <- cm_rfw$byClass["Sensitivity"]
  #
  # #specificity
  # err_mat_spec[i,4] <- cm_rfw$byClass["Specificity"]
  #
  #
  # #########
  # # BALANCED RANDOM FOREST - not prepped yet
  # #########
  #
  # nmin = sum(kiva_tr$Class =='bad') # minority class is Yes
  #
  # #also have 100 trees
  # rfb = randomForest(Class ~ .,
  #                         kiva_tr,
  #                         ntree = 100,
  #                         sample_size = rep(nmin, 2),
  #                         trControl = fitControl,
  #                         metric = "ROC",
  #                         tuneGrid = grid_mtry)
  # #predict
  # pred_rfb <- predict(rfb, kiva_te[-35])
  #
  # #calculate errors and save to matrices
  #
  # #accuracy
  # err_mat_accuracy[i,5] <- 1 - mean(pred_rfb != kiva_te[,35])
  #
  # #calc confusion
  # cm_rfb <- confusionMatrix(pred_rfb, kiva_te[[35]],
  #                           positive = "good")
  #
  # #sensitivity
  # err_mat_sens[i,5] <- cm_rfb$byClass["Sensitivity"]
  #
  # #specificity
  # err_mat_spec[i,5] <- cm_rfb$byClass["Specificity"]


} #END OF LOOP!!
```

```
#prep the data to compare results

#accuracy
colnames(err_mat_accuracy) <- c("Base RF",
                                "Repayment Interval RF",
                                "Country RF",
                                "All RF")

accuracydf <- err_mat_accuracy %>% gather(ClassificationType, Accuracy)

#sensitivity
colnames(err_mat_sens) <- c("Base RF",
                                "Repayment Interval RF",
                                "Country RF",
                                "All RF")

sensitivitydf <- err_mat_sens %>% gather(ClassificationType, Sensitivity)


#specificity
colnames(err_mat_spec) <- c("Base RF",
                                "Repayment Interval RF",
                                "Country RF",
                                "All RF")

specificitydf <- err_mat_spec %>% gather(ClassificationType, Specificity)

#put all of the dfs together
metricsdf <- left_join(accuracydf, sensitivitydf) %>% left_join(specificitydf)
metricsdf
```

Tune the models that might actually be able to make some predictions

```
imp_rf_all[[1]]
```

```
#prepare the data for graphing

#accuracy
colnames(err_mat_accuracy) <- c("Base RF",
                                "Repayment Interval RF",
                                "Country RF",
                                "All RF")

accuracydf <- err_mat_accuracy %>% gather(ClassificationType, Accuracy)

#sensitivity
colnames(err_mat_sens) <- c("Base RF",
                                "Repayment Interval RF",
                                "Country RF",
                                "All RF")

sensitivitydf <- err_mat_sens %>% gather(ClassificationType, Sensitivity)


#specificity
colnames(err_mat_spec) <- c("Base RF",
                                "Repayment Interval RF",
                                "Country RF",
                                "All RF")

specificitydf <- err_mat_spec %>% gather(ClassificationType, Specificity)


#make the graphs -- use the same y scale for better comparison
ggplot(
  data = accuracydf,
  aes(x = ClassificationType, y = Accuracy)
) + geom_boxplot() +
  scale_y_continuous(limits = c(0, 1)) +
  ggtitle("Accuracy by Type")

ggplot(
  data = sensitivitydf,
  aes(x = ClassificationType, y = Sensitivity)
) + geom_boxplot() +
  scale_y_continuous(limits = c(0, 1)) +
  ggtitle("Sensitivity by Type")


ggplot(
  data = specificitydf,
  aes(x = ClassificationType, y = Specificity)
) + geom_boxplot() +
  scale_y_continuous(limits = c(0, 1)) +
  ggtitle("Specificity by Type")
```