FinalProject\_IS457\_35

35

April 29, 2019

# Load libraries  
  
# install.packages("ggplot2")  
# install.packages("dplyr")  
# install.packages("lattice")  
# install.packages("reshape2")  
  
library(ggplot2)  
library(dplyr)  
library(lattice)  
library(reshape2)  
#require(RColorBrewer)

PART 1: Data Processing

# Q1

**1.1: What variables have missing values? What types/forms of missing values are they?**

# Initial read of the data:  
# airbnb <- read.csv(paste(getwd(),"/data/AirbnbSydney.csv", sep = ""), stringsAsFactors = FALSE)  
  
# Read in airbnb data -- accounting for NA values  
airbnb <- read.csv(paste(getwd(),"/data/AirbnbSydney.csv", sep = ""), stringsAsFactors = FALSE, na.strings = c("N/A","", "NA"))  
  
  
# Number of missing values  
missingvals <- sapply(airbnb, function(x) sum(is.na(x)))  
missingvals[missingvals>0]

## neighborhood\_overview house\_rules   
## 664 1639   
## host\_response\_time host\_response\_rate   
## 2483 2483   
## city zipcode   
## 8 21   
## bathrooms bedrooms   
## 1 1   
## cleaning\_fee review\_scores\_rating   
## 621 1   
## review\_scores\_accuracy review\_scores\_cleanliness   
## 1 1   
## review\_scores\_checkin review\_scores\_communication   
## 1 1

# Percent missing values  
round(missingvals[missingvals >0]/length(airbnb[,1]), 2)

## neighborhood\_overview house\_rules   
## 0.06 0.15   
## host\_response\_time host\_response\_rate   
## 0.23 0.23   
## city zipcode   
## 0.00 0.00   
## bathrooms bedrooms   
## 0.00 0.00   
## cleaning\_fee review\_scores\_rating   
## 0.06 0.00   
## review\_scores\_accuracy review\_scores\_cleanliness   
## 0.00 0.00   
## review\_scores\_checkin review\_scores\_communication   
## 0.00 0.00

summary(airbnb[,missingvals>0])

## neighborhood\_overview house\_rules host\_response\_time  
## Length:10815 Length:10815 Length:10815   
## Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character   
##   
##   
##   
##   
## host\_response\_rate city zipcode bathrooms   
## Length:10815 Length:10815 Length:10815 Min. : 0.000   
## Class :character Class :character Class :character 1st Qu.: 1.000   
## Mode :character Mode :character Mode :character Median : 1.000   
## Mean : 1.349   
## 3rd Qu.: 1.500   
## Max. :10.000   
## NA's :1   
## bedrooms cleaning\_fee review\_scores\_rating  
## Min. : 0.000 Length:10815 Min. : 20.00   
## 1st Qu.: 1.000 Class :character 1st Qu.: 92.00   
## Median : 1.000 Mode :character Median : 96.00   
## Mean : 1.629 Mean : 94.19   
## 3rd Qu.: 2.000 3rd Qu.:100.00   
## Max. :14.000 Max. :100.00   
## NA's :1 NA's :1   
## review\_scores\_accuracy review\_scores\_cleanliness review\_scores\_checkin  
## Min. : 2.00 Min. : 2.000 Min. : 2.000   
## 1st Qu.: 9.00 1st Qu.: 9.000 1st Qu.:10.000   
## Median :10.00 Median :10.000 Median :10.000   
## Mean : 9.64 Mean : 9.398 Mean : 9.782   
## 3rd Qu.:10.00 3rd Qu.:10.000 3rd Qu.:10.000   
## Max. :10.00 Max. :10.000 Max. :10.000   
## NA's :1 NA's :1 NA's :1   
## review\_scores\_communication  
## Min. : 2.000   
## 1st Qu.:10.000   
## Median :10.000   
## Mean : 9.802   
## 3rd Qu.:10.000   
## Max. :10.000   
## NA's :1

#sapply(airbnb[,missingvals>0], function(x) str(x))  
  
# Clean the prices (drop $ and comma, as.numeric)  
airbnb$price <- as.numeric(gsub("^\\$|,","",airbnb$price))  
airbnb$cleaning\_fee <- as.numeric(gsub("^\\$|,","",airbnb$cleaning\_fee))  
airbnb$extra\_people <- as.numeric(gsub("^\\$|,","",airbnb$extra\_people))  
  
# Clean percents  
airbnb$host\_response\_rate <- as.numeric(gsub("%$","", airbnb$host\_response\_rate))  
  
# Store host\_since as a date  
airbnb$host\_since <- as.Date(airbnb$host\_since, format = "%m/%d/%y") # add column of day units for comparing ages  
  
# Add a column -- number of days as host as of max(date)  
airbnb$host\_number\_of\_days <- difftime(max(airbnb$host\_since), airbnb$host\_since, units = "days")  
  
  
# Store logicals as logical. R requires a capial T or F  
airbnb$host\_is\_superhost <- as.logical(toupper(airbnb$host\_is\_superhost))  
airbnb$host\_identity\_verified <- as.logical(toupper(airbnb$host\_identity\_verified))

### My answer

* Neighborhood\_overview and house\_rules are both text-heavy (paragraph or more) fields, NA on 6 and 15%. NA values are shown as empty strings: ""
* Host\_response\_time and host\_response\_rate are both absent from 23% of the observations. Response Time is an ordinal categorization of how long it takes to respond. Response Rate is a percentage that is imported as character, but will be converted to numeric. NA values are shown as valid character element “N/A”.
* City and zipcode are address components useful for aggregation. They are missing from 8 and 21 records (nearly zero percent) of these character vectors.
* Bathrooms is a numeric with decimals (due to half bath, .5). NA values are characters “NA”.
* Bedrooms is an integer numeric. NA values are characters “NA”.
* Cleaning\_fee is a numerical (after cleaning) and missing from 6% of records.
* Review\_scores\_rating and review\_scores\_xxx are missing one observation in each column. These are integer values. NA values are characters “NA”.

# 1.2: How will you deal with missing values? Justify your methods.

For categorical data with missing values (such as city, zipcode, response\_time), I imputed a value of “unknown.” This preserves the known data, and allows for omission of this value if necessary. I find it preferable to reduce the number of observations than to potentially impute incorrect values.

For numerical data, I used the median. This value is not affected by outliers. It is simply the value where half of the observations are higher, and half are lower.

# 1.3: Describe how your choice method may impact later analysis.

Imputing “unknown” for categorical data adds another category. It helps to preserve the known values for that observation, so na.rm() won’t remove the observation with the analyst possibly being unaware of its removal.

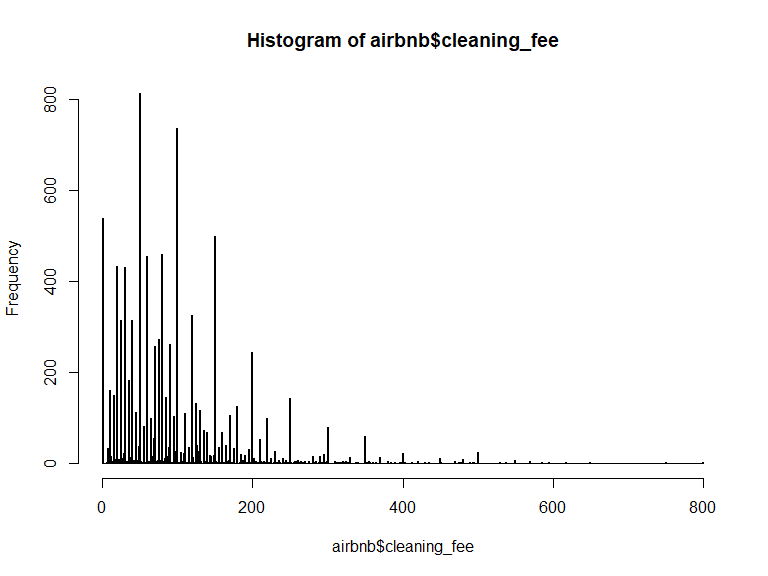
Imputing median values will increase the measure of centrality, while reducing the variance and standard deviation for those variables.

# 1.4: Implement methods to deal with missing values.

# NA neighborhood\_overview and house\_rules  
# Do nothing, because these are descriptive text values and there is no way to impute them.  
  
getmode <- function(v){  
 # funcname: getmode  
 # inputs : a vector of elements  
 # outputs : a single-value vector containing the most-frequently occuring value  
 # purpose : Calculate the mode  
 # related : mean(), median()  
 # auth/dt : ID35,   
   
 uniqv <- unique(v)  
 return(uniqv[which.max(tabulate(match(v, uniqv)))])  
}  
  
# NA host\_response\_time  
# To preserve the integrity of these categorical data for comparative purposes,  
# I will create a new category for NA values called "unknown"  
airbnb$host\_response\_time[is.na(airbnb$host\_response\_time)] <- "unknown"  
  
# NA host\_response\_rate  
airbnb$host\_response\_rate[is.na(airbnb$host\_response\_rate)] <- median(airbnb$host\_response\_rate, na.rm = TRUE)  
  
# NA city  
airbnb$city[is.na(airbnb$city)] <- "unknown"  
  
# NA zipcode  
airbnb$zipcode[is.na(airbnb$zipcode)] <- "unknown"  
  
# NA bathrooms  
airbnb$bathrooms[is.na(airbnb$bathrooms)] <- median(airbnb$bathrooms, na.rm = TRUE)  
  
# NA bedrooms  
# Description calls this one a studio (no bedroom)  
airbnb$bedrooms[is.na(airbnb$bedrooms)] <- 0  
  
# NA cleaning fee  
# The mode is 50  
getmode(airbnb$cleaning\_fee)

## [1] 50

# The distribution is positively skewed. This explains why the mean (94.4) is greater than the median (80)  
hist(airbnb$cleaning\_fee, breaks = 1000)



# To clean this, I will use median. It lies between the mode and the mean, so I'm using the central measure of centrality. :)  
airbnb$cleaning\_fee[is.na(airbnb$cleaning\_fee)] <- median(airbnb$cleaning\_fee, na.rm = TRUE)  
  
  
# NA review scores  
# The review scores (review\_scores\_rating, review\_scores\_accuracy,   
# review\_scores\_cleanliness, review\_scores\_checkin, and review\_scores\_communication)  
# are each missing one value. I will use median as the measure of centrality  
# to fill in the missing values.  
  
# These columns are being handled in the same way, and are next to each other.  
# I will use a for loop to go through each column, and assign the median value  
# of that column to any NA values.  
for(i in 28:34){  
 airbnb[is.na(airbnb[,i]),i] <- median(airbnb[,i], na.rm = TRUE)  
}

# 1.5: After dealing with missing values, show the dimensions of the data.

# Show dimensions  
dim(airbnb)

## [1] 10815 37

# 1.6: Comment on and explain any other data cleaning or preparation steps you think would be necessary from your inspection of the data (you do not have to carry them out).

host\_since will need to be converted to date. host\_response\_time should become a factor (to compare levels) host\_response\_rate need the % sign scrubbed and converted to numeric (maybe even percentage, e.g. /100) price and cleaning\_fee need $ scrubbed and converted to numeric

# Q2

**Conduct a preliminary exploration and describe what you find interesting or unexpected.**

# This will create two lists: counts (table()) for categorical data, and summaries for continuous data  
# This allows me to visually inspect the numerical distribution of each variable (using View())  
col\_categories <- c(7,9,11:20,35)  
col\_continuous <- c(6,8,22:34,36)  
  
counts <- lapply(airbnb[,col\_categories], function(x) table(x))  
summaries <- lapply(airbnb[,col\_continuous], function(x) summary(x))  
sds <- lapply(airbnb[,col\_continuous], function(x) sd(x))  
  
counts

## $host\_response\_time  
## x  
## a few days or more unknown within a day   
## 119 2483 1078   
## within a few hours within an hour   
## 1386 5749   
##   
## $host\_is\_superhost  
## x  
## FALSE TRUE   
## 8020 2795   
##   
## $host\_identity\_verified  
## x  
## FALSE TRUE   
## 5631 5184   
##   
## $city  
## x  
## â\200¢ Darling harbour Abbotsford   
## 1 8   
## æ‚‰å°¼ Agnes Banks   
## 1 1   
## Alexandria Alexandria   
## 83 1   
## Allambie Heights Allawah   
## 11 3   
## Allawah/Carlton Annandale   
## 1 75   
## Arcadia Arncliffe   
## 2 43   
## Artarmon Ashbury, Sydney   
## 7 1   
## Ashfield Ashfield, New South Wales, AU   
## 46 1   
## Asquith Auburn   
## 1 7   
## Auburn Auburn / Lidcomb   
## 1 1   
## Avalon Avalon Beach   
## 21 118   
## Balgowlah Balgowlah Heights   
## 56 12   
## Balmain Balmain / Birchgrove   
## 79 1   
## Balmain East Balmoral Beach   
## 18 2   
## Bangor Banksia   
## 2 3   
## Banksia Sydney Bankstown   
## 1 22   
## Bar Point Barangaroo   
## 1 2   
## Bardia Bardwell Valley   
## 2 3   
## Barpoint Baulkham Hills   
## 1 7   
## Bayview Beacon Hill   
## 6 14   
## Beaconsfield Beaumont Hills   
## 13 2   
## Beecroft Belfield   
## 4 3   
## Bella Vista Bellevue Hill   
## 7 67   
## Bellevue Hill (Double Bay side). Bellevue Hill, Sydney   
## 1 1   
## Belmore Berala   
## 4 10   
## Berowra Creek Berowra Heights   
## 1 2   
## Berowra Waters Beverly Hills   
## 3 5   
## Bexley Bexley North   
## 10 3   
## Bilgola Bilgola Beach   
## 2 17   
## Bilgola Plateau Bilgola, Sydney   
## 16 1   
## Birchgrove Blacktown   
## 34 4   
## Blair Athol Blakehurst   
## 1 2   
## Bondi Bondi   
## 222 2   
## bondi beach Bondi Beach   
## 1 555   
## Bondi beach Bondi Beach, Sydney   
## 2 2   
## Bondi Junction Bondi Junction   
## 142 1   
## Bondi Junction Sydney Bondi Junction, Sydney   
## 1 2   
## Bondi, Tamarama Botany   
## 1 14   
## Breakfast Point Brighton-Le-Sands   
## 1 27   
## Brighton Le Sands Bronte   
## 1 149   
## Bronte Brooklyn   
## 1 2   
## Brookvale Bundeena   
## 3 25   
## Bungarribee Burraneer   
## 1 1   
## Burraneer/Cronulla Burwood   
## 1 35   
## Cabarita Cabramatta   
## 1 3   
## Cabramatta West Cammeray   
## 1 28   
## Campbelltown camperdown   
## 2 1   
## Camperdown Campsie   
## 88 10   
## Canada Bay Canley Heights   
## 1 2   
## Canterbury Caringbah   
## 8 1   
## Caringbah South Carlingford   
## 3 9   
## Carlton Carnes Hill   
## 6 9   
## Carramar Carss Park   
## 6 2   
## Castle Cove Castle Hill   
## 6 7   
## Castlecrag Casula   
## 5 8   
## Centennial Park Centennial Park, Sydney   
## 13 1   
## Chatswood Chatswood Sydney   
## 50 1   
## Chatswood West Chatswood, Sydney   
## 4 1   
## Cheltenham Cherrybrook   
## 3 3   
## Chester Hill Chifley   
## 4 4   
## Chippendale Chippendale   
## 155 1   
## Chiswick Church Point   
## 8 4   
## Clareville Clontarf   
## 8 11   
## Clovelly Coasters Retreat   
## 72 3   
## Collaroy Collaroy Beach   
## 16 1   
## Collaroy Plateau Como   
## 5 1   
## Concord Concord West   
## 4 2   
## Connells Point Coogee   
## 5 272   
## Coogee Coogee, New South Wales, Australia   
## 1 1   
## Cottage Point Council of the City of Sydney   
## 2 6   
## Cremorne Cremorne Point   
## 48 13   
## Cromer Cronulla   
## 5 50   
## Crows Nest Crows Nest, Sydney   
## 31 1   
## Croydon Croydon Park   
## 10 7   
## Croydon Park NSW Curl Curl   
## 1 10   
## Daceyville Dangar Island   
## 2 4   
## darling harbour Darling Point   
## 3 31   
## Darlinghurst Darlinghurst   
## 373 1   
## Darlinghurst Sydney Darlinghurst, Sydney   
## 1 1   
## Darlington Davidson   
## 32 2   
## Dawes Point Dee Why   
## 6 46   
## Dee Why, Sydney Denistone   
## 1 2   
## Denistone East Dolls Point   
## 4 3   
## Double Bay Double Bay, Sydney   
## 43 1   
## Dover Heights Drummoyne   
## 15 22   
## Duffys Forest Dulwich Hill   
## 2 34   
## Dundas Dundas Valley   
## 1 4   
## Dural Eagle Vale   
## 3 1   
## Earlwood East Hills   
## 15 2   
## East Lindfield Eastern Creek   
## 3 1   
## Eastgardens Eastgardens   
## 9 1   
## Eastlakes Eastwood   
## 9 10   
## Edgecliff Edmondson Park   
## 25 3   
## Elanora Heights Elderslie   
## 4 4   
## Elizabeth Bay Elizabeth Bay   
## 101 1   
## Elizabeth Bay / Sydney Elizabeth Bay /Potts Point   
## 1 1   
## Emu Plains Engadine   
## 1 1   
## Enmore Epping   
## 24 14   
## Ermington Erskineville   
## 8 87   
## Eveleigh Fairfield   
## 4 1   
## Fairfield West Fairlight   
## 1 83   
## Fairlight Fairlight (Manly)   
## 1 1   
## Five Dock Forest lodge   
## 7 1   
## Forest Lodge Forestville   
## 50 8   
## Frenchs Forest Freshwater   
## 7 59   
## Gladesville Glebe   
## 8 93   
## Glenfield Glenhaven   
## 18 1   
## Glenmore Park Glenorie   
## 1 1   
## Glenwood Gordon   
## 1 6   
## Granville Grays Point   
## 3 1   
## Great Mackerel Beach Greenacre   
## 9 2   
## Greenhills Beach Greenwich   
## 2 19   
## Greystanes Greystanes   
## 6 1   
## Guildford Guildford West   
## 5 3   
## Gymea Haberfield   
## 1 8   
## Harris Park Haymarket   
## 1 92   
## Heathcote Henley   
## 1 3   
## Hillsdale Holroyd   
## 10 1   
## Holsworthy Homebush   
## 3 14   
## Homebush West Horningsea Park   
## 16 1   
## Hornsby Hornsby Heights   
## 8 1   
## Hunters Hill Huntleys Cove   
## 12 1   
## Hurlstone Park Hurstville   
## 8 18   
## Hurstville Hurstville Grove   
## 1 1   
## Hurstville Sydney Ingleburn   
## 1 2   
## Ingleside Jannali   
## 2 1   
## Jordan Springs Kellyville   
## 1 3   
## Kensington Killara   
## 35 6   
## Killarney Heights Kings Langley   
## 4 4   
## Kingsford Kingsgrove   
## 35 2   
## Kingswood Kirkham   
## 3 1   
## Kirribilli Kogarah   
## 49 18   
## Kogarah Bay Kurnell   
## 3 2   
## Kurraba Point Kyeemagh   
## 5 2   
## Kyle Bay La Perouse   
## 1 1   
## Lane Cove Lane Cove North   
## 18 24   
## Lane Cove West Lansvale   
## 8 1   
## Lavender Bay Leets Vale   
## 11 1   
## Leichhardt Leichhardt Municipal Council   
## 76 1   
## Leonay lewisham   
## 2 1   
## Lewisham Liberty Grove   
## 22 5   
## Lidcombe Lilli Pilli   
## 14 1   
## Lilyfield Lindfield   
## 32 6   
## Linley Point Little bay   
## 1 1   
## Little Bay Liverpool   
## 17 3   
## Loftus Longueville   
## 1 9   
## Lovett Bay Lower Portland   
## 3 2   
## Luddenham Lugarno   
## 2 1   
## Macquarie Park Maianbar   
## 14 5   
## Malabar Manly   
## 17 389   
## Manly Manly Beach   
## 1 4   
## Manly Vale Maroubra   
## 26 139   
## Maroubra Beach Maroubra, New South Wales, AU   
## 1 1   
## Marrickville Marrickville   
## 90 1   
## Marsden Park Marsfield   
## 3 12   
## Mascot Matraville   
## 95 6   
## Mccarrs Creek McMahons Point   
## 1 24   
## Meadowbank Merrylands   
## 3 11   
## Merrylands West Middle Dural   
## 4 3   
## Millers Point Millers Point, Sydney   
## 39 1   
## Milsons Passage Milsons Point   
## 4 16   
## Miranda Mona Vale   
## 3 30   
## Monterey Moorebank   
## 5 1   
## Morning Bay Mortdale   
## 4 2   
## Mortlake Mosman   
## 2 167   
## Mosman Sydney Mount Annan   
## 1 2   
## Mount Colah Mount Druitt   
## 1 1   
## Mount Pritchard Naremburn   
## 1 18   
## Narrabeen Narraweena   
## 24 1   
## Neutral Bay New South Wales   
## 62 1   
## New South Wales Newington   
## 1 10   
## Newport Newtown   
## 49 190   
## North Balgowlah North Bondi   
## 25 191   
## North Curl Curl North Manly   
## 16 14   
## North Narrabeen North Parramatta   
## 17 8   
## North Ryde North St Marys   
## 17 2   
## North Strathfield North Sydney   
## 1 65   
## North Sydney North Sydney / Kirribilli   
## 1 1   
## North Sydney / Waverton North Sydney Council   
## 1 1   
## North Willoughby Northbridge   
## 6 6   
## Northern Beaches Northmead   
## 1 3   
## Northwood NSW   
## 2 1   
## Oatlands Oatley   
## 2 3   
## Oran Park Orchard Hills   
## 1 1   
## Oxford Falls Oxley Park   
## 2 5   
## Paddington Padstow   
## 175 15   
## Padstow Heights Pagewood   
## 1 10   
## Palm Beach Panania   
## 57 6   
## Parramatta Peakhurst   
## 30 3   
## Peakhurst Heights Pemulwuy   
## 1 2   
## Pennant Hills Penrith   
## 4 7   
## Penshurst Petersham   
## 5 23   
## Phillip Bay Pittwater Council   
## 1 1   
## Plumpton Point Piper   
## 1 1   
## Port Jackson Potts Point   
## 1 198   
## Potts Point Potts Point, New South Wales, AU   
## 1 1   
## Prestons Prospect   
## 2 1   
## Punchbowl Putney   
## 1 7   
## Pymble pyrmont   
## 4 1   
## Pyrmont Pyrmont   
## 178 2   
## Quakers Hill Queens park   
## 1 1   
## Queens Park Queenscliff   
## 21 35   
## Ramsgate Randwick   
## 1 221   
## Redfern Revesby   
## 206 4   
## Revesby Heights Rhodes   
## 1 36   
## Rhodes Riverview   
## 1 2   
## Riverwood Rockdale   
## 18 9   
## Rose Bay Rose Bay, Sydney   
## 69 1   
## Rosebery Rosehill   
## 59 2   
## Roseville Rossmore   
## 20 1   
## Rouse Hill Rozelle   
## 3 66   
## Rozelle / Balmain Rozelle, Sydney   
## 1 1   
## Rushcutters Bay Russell Lea   
## 64 6   
## Rydalmere Ryde   
## 1 25   
## Saint Clair Saint Ives   
## 1 3   
## Saint Ives Chase Saint Leonards   
## 1 4   
## Saint Marys Saint Peters   
## 1 14   
## Sandringham Sans Souci   
## 8 19   
## Scotland Island Seaforth   
## 11 18   
## Seven Hills Seven Hills   
## 1 1   
## Smithfield South Coogee   
## 1 17   
## South Hurstville South Wentworthville   
## 1 6   
## St Ives St Leonards   
## 2 17   
## St Peters Stanhope Gardens   
## 8 2   
## Stanmore Strathfield   
## 33 35   
## Strathfield South Summer Hill   
## 3 16   
## Surry hills Surry Hills   
## 2 500   
## Surry Hills Sutherland   
## 1 2   
## Sydenham sydney   
## 8 3   
## Sydney Sydney   
## 463 8   
## Sydney CBD Sydney City   
## 2 1   
## Sydney Olympic Park Sydney, Bondi Beach   
## 89 1   
## Sylvania Tamarama   
## 3 72   
## Taren Point Telopea   
## 1 4   
## Tempe Terrey Hills   
## 11 1   
## The Ponds The Rocks   
## 3 14   
## Thornleigh Toongabbie   
## 1 3   
## Turramurra Turrella   
## 5 4   
## Ultimo Ultimo   
## 115 1   
## unknown Vaucluse   
## 8 46   
## Wahroonga Waitara   
## 2 2   
## Wareemba Warrawee   
## 6 1   
## Warriewood Warwick Farm   
## 9 2   
## Waterfall waterloo   
## 1 1   
## Waterloo Watsons Bay   
## 131 6   
## Waverley Waverton   
## 52 12   
## Wentworth point Wentworth Point   
## 1 22   
## Werrington Downs West Hoxton   
## 1 2   
## West Pennant Hills West Pymble   
## 6 1   
## West Ryde Westmead   
## 4 5   
## Whale Beach Wheeler Heights   
## 11 1   
## Wiley Park Willoughby   
## 1 14   
## Willoughby East Winston Hills   
## 5 1   
## Wisemans Ferry Wolli Creek   
## 1 43   
## Wolli Creek Wollstonecraft   
## 1 26   
## Wollstonecraft, Sydney Woodbine   
## 1 1   
## Woollahra Woolloomooloo   
## 63 60   
## Woolloomooloo Woolooware   
## 1 2   
## Woolwich Yagoona   
## 3 2   
## Yowie Bay Zetland   
## 1 93   
##   
## $zipcode  
## x  
## 2000 2007 2008 2009 2010 2011 2015 2016   
## 551 127 187 183 876 442 102 210   
## 2017 2018 2019 2020 2021 2022 2023 2024   
## 228 70 14 96 189 169 69 201   
## 2025 2026 2027 2028 2029 2030 2031 2032   
## 64 1056 59 46 70 69 296 37   
## 2033 2034 2035 2036 2037 2038 2039 2040   
## 36 290 157 68 143 76 69 109   
## 2041 2042 2043 2044 2045 2046 2047 2048   
## 134 215 85 43 8 36 22 33   
## 2049 2050 2060 2061 2062 2063 2064 2065   
## 46 88 116 65 26 6 7 119   
## 2066 2067 2068 2069 2070 2071 2072 2073   
## 64 56 30 26 9 6 6 5   
## 2074 2075 2076 2077 2079 2082 2083 2084   
## 6 6 3 12 1 6 12 5   
## 2085 2086 2087 2088 2089 2090 2091 2092   
## 2 7 12 173 69 62 1 18   
## 2093 2094 2095 2096 2097 2099 2100 2101   
## 130 87 392 105 24 70 45 47   
## 2102 2103 2104 2105 2106 2107 2108 2110   
## 9 30 6 23 49 194 70 15   
## 2111 2112 2113 2114 2115 2116 2117 2118   
## 12 35 31 9 8 1 11 9   
## 2119 2120 2121 2122 2125 2126 2127 2130   
## 7 5 14 22 6 3 123 16   
## 2131 2132 2133 2134 2135 2136 2137 2138   
## 49 10 8 35 35 3 9 45   
## 2140 2141 2142 2144 2145 2146 2147 2148   
## 30 25 6 8 20 3 6 5   
## 2150 2151 2152 2153 2154 2155 2156 2157   
## 30 9 4 15 7 8 1 1   
## 2158 2159 2160 2161 2162 2163 2164 2165   
## 6 2 15 8 4 6 1 2   
## 2166 2167 2170 2171 2173 2174 2190 2191   
## 7 18 17 12 3 3 2 3   
## 2192 2193 2194 2195 2196 2199 2200 2203   
## 4 17 10 1 1 2 22 35   
## 2204 2205 2206 2207 2208 2209 2210 2211   
## 93 92 15 16 2 5 24 16   
## 2212 2213 2216 2217 2218 2219 2220 2221   
## 5 8 43 27 10 30 21 11   
## 2222 2223 2224 2226 2227 2228 2229 2230   
## 4 5 3 2 1 4 6 87   
## 2231 2232 2233 2234 2557 2558 2560 2565   
## 2 4 3 2 1 1 4 4   
## 2567 2570 2745 2747 2748 2750 2753 2756   
## 2 6 3 5 1 10 1 2   
## 2759 2760 2761 2762 2763 2765 2766 2767   
## 1 8 1 1 1 3 1 1   
## 2768 2769 2770 2775 NSW 2025 unknown   
## 3 3 1 2 1 21   
##   
## $property\_type  
## x  
## Aparthotel Apartment Bed and breakfast   
## 2 6222 46   
## Boat Boutique hotel Bungalow   
## 8 26 62   
## Cabin Camper/RV Campsite   
## 35 2 2   
## Chalet Condominium Cottage   
## 1 309 59   
## Dome house Farm stay Guest suite   
## 1 3 258   
## Guesthouse Hostel Hotel   
## 200 46 10   
## House Hut Island   
## 2604 1 3   
## Loft Other Serviced apartment   
## 150 16 47   
## Tent Tiny house Tipi   
## 2 15 1   
## Townhouse Treehouse Villa   
## 589 1 93   
## Yurt   
## 1   
##   
## $room\_type  
## x  
## Entire home/apt Private room Shared room   
## 7922 2809 84   
##   
## $accommodates  
## x  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15   
## 556 4200 766 2350 669 1208 251 430 84 163 31 49 12 19 13   
## 16   
## 14   
##   
## $bathrooms  
## x  
## 0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 7 8   
## 13 19 7528 569 1881 334 307 59 64 16 10 3 9 1 1   
## 10   
## 1   
##   
## $bedrooms  
## x  
## 0 1 2 3 4 5 6 7 14   
## 682 5475 2819 1107 529 164 30 8 1   
##   
## $beds  
## x  
## 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14   
## 69 4852 2623 1483 877 443 227 98 72 22 27 5 6 3 2   
## 16 18 29   
## 4 1 1   
##   
## $bed\_type  
## x  
## Airbed Couch Futon Pull-out Sofa Real Bed   
## 11 8 12 46 10738   
##   
## $cancellation\_policy  
## x  
## flexible moderate   
## 1391 3314   
## strict\_14\_with\_grace\_period super\_strict\_30   
## 6089 1   
## super\_strict\_60   
## 20

* host\_response\_time: Most hosts respond quickly (within an hour), although the imputed unknown category makes up the second-largest percentage.
* host\_is\_superhost: 2795 of the 10,815 are superhosts.
* host\_identity\_verified: just under half of listing have a verified host. This even number could make for good comparisons if splitting the data.
* city: What a mess! Invalid charactersets, inconsistent naming and upper/lower casing of city names (see: Bondi Beach, bondi beach, Bondi beach, “Bondi Beach, Sydney”). I will ignore if possible.
* zipcode: The distribution is uneven, and I don’t know anything about the districts themselves. However, it’s the cleanest geographic data available.
* property\_type: Mostly Apartment (~58%) and House (~24%), although 31 property types total.
* room\_type: Three categories. Predominantly Entire home/apt, 26% Private room, and only 84 in a Shared room.
* accommodates: Mostly 2, 4, 6, 3, 1, 5, then 8, 7, 10, and up to 16. Even numbers are more common.
* bathrooms: Mostly 1 or 2, but goes up to 10!
* bedrooms: Zero to four are “common” (>500), but continous up to 7 BR, then an outlier with 14 bedrooms.
* beds: Zero to 14, then 16, 18, and 29(!). Interesting about zero beds; maybe a couch?
* bed\_type: Aha, couch is a type. 99.28% Real bed, with the remainder scattered around lesser bed types. There are 77 non-Real bed observations. It is plausible that the 69 zero beds observations are included in the 77.
* cancellation\_policy: Five categories – flexible, moderate, strict and two levels of super\_strict. I wonder if this affects reviews?

summaries

## $host\_since  
## Min. 1st Qu. Median Mean 3rd Qu.   
## "2009-04-20" "2013-11-08" "2015-02-01" "2015-02-04" "2016-05-29"   
## Max.   
## "2018-11-25"   
##   
## $host\_response\_rate  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 100.00 100.00 97.23 100.00 100.00   
##   
## $price  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 96.0 150.0 203.2 230.0 10001.0   
##   
## $cleaning\_fee  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 40.00 80.00 93.64 120.00 800.00   
##   
## $guests\_included  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 1.000 1.969 2.000 16.000   
##   
## $extra\_people  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 0.00 10.00 17.07 25.00 410.00   
##   
## $minimum\_nights  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 2.000 4.078 3.000 500.000   
##   
## $number\_of\_reviews  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 4.00 12.00 28.94 36.00 493.00   
##   
## $review\_scores\_rating  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 20.00 92.00 96.00 94.19 100.00 100.00   
##   
## $review\_scores\_accuracy  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.00 9.00 10.00 9.64 10.00 10.00   
##   
## $review\_scores\_cleanliness  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 9.000 10.000 9.398 10.000 10.000   
##   
## $review\_scores\_checkin  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 10.000 10.000 9.782 10.000 10.000   
##   
## $review\_scores\_communication  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 10.000 10.000 9.802 10.000 10.000   
##   
## $review\_scores\_location  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 10.000 10.000 9.737 10.000 10.000   
##   
## $review\_scores\_value  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 9.000 10.000 9.385 10.000 10.000   
##   
## $reviews\_per\_month  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.020 0.270 0.950 1.572 2.310 15.180

sds

## $host\_since  
## [1] 681.2154  
##   
## $host\_response\_rate  
## [1] 11.2011  
##   
## $price  
## [1] 254.7939  
##   
## $cleaning\_fee  
## [1] 77.63737  
##   
## $guests\_included  
## [1] 1.584671  
##   
## $extra\_people  
## [1] 25.77062  
##   
## $minimum\_nights  
## [1] 12.66408  
##   
## $number\_of\_reviews  
## [1] 42.66588  
##   
## $review\_scores\_rating  
## [1] 7.471855  
##   
## $review\_scores\_accuracy  
## [1] 0.7430775  
##   
## $review\_scores\_cleanliness  
## [1] 0.9561665  
##   
## $review\_scores\_checkin  
## [1] 0.6092216  
##   
## $review\_scores\_communication  
## [1] 0.6054712  
##   
## $review\_scores\_location  
## [1] 0.5801447  
##   
## $review\_scores\_value  
## [1] 0.8401076  
##   
## $reviews\_per\_month  
## [1] 1.744816

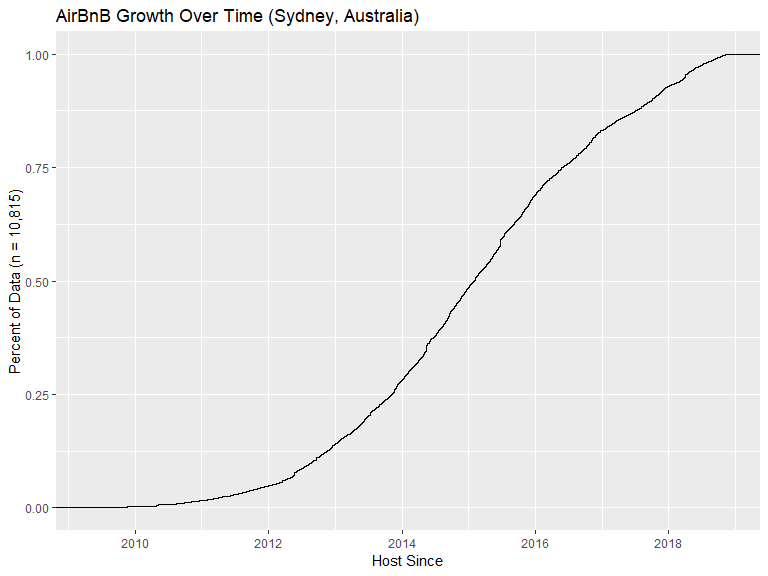
Summaries help with quartiles and ranges, skew can be somewhat interpreted by comparing median to mean.

* host\_since: median and mean almost identical; six years from min to median, 3.5 years median to max. Listings have been added more frequently over time.
* host\_response\_rate: 1st quartile is 100%, so most hosts respond to inquiries.
* price: median of 150 and mean of 203 indicates positive skew. This is expected on this type of variable; prices have no upper limit, but are bound on the lower end by zero.
* cleaning\_fee: same positive skew as price
* guests\_included: strong positive skew. median is 1, 3rd quar is 2, and mean is 1.969. Max of 16
* extra\_people: the max value of 410 may be influencing the mean
* minimum\_nights: skewed HEAVILY by the max of 500. A 500 night minimum seems to go against the typical use case for AirBnB that I understand. Higher minimum night values will suppress the possible number of reviews per month. I want to explore this.
* number\_of\_reviews: median is 12; max is 493. Indicates popularity and longevity of a listing.
* review\_scores\_rating: the overall rating on a scale of 20 to 100 (the minimum is 20). Can I assume this relates to a 5 star rating scale, where each star is worth 20 points?
* review\_scores\_xxx: the xxx refers to subcategories of six review components: accuracy (of listing description), cleanliness, checkin, communication, location, and value.
* reviews\_per\_month: effectively – number\_of\_reviews / number of months listed. median .95, mean 1.5. max of 15.18 (averaging a new guest every other night!)

# Q3

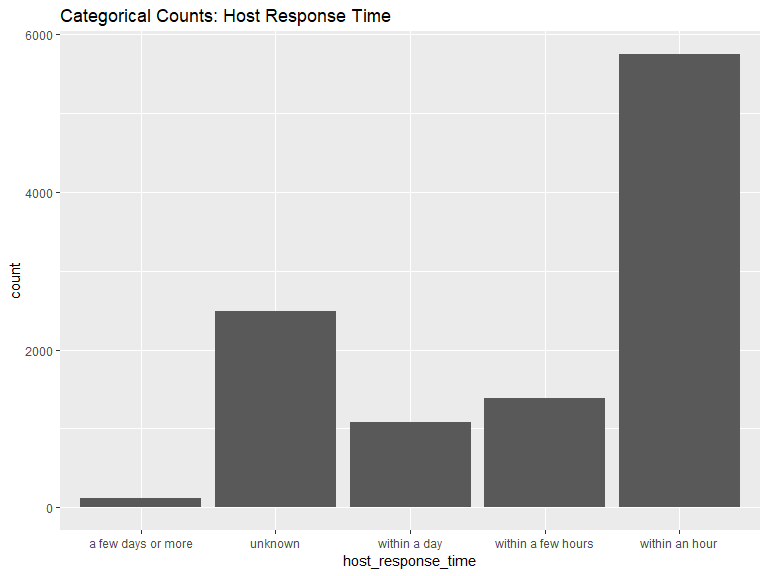
**Explore comprehensively with charts, tables, and graphs** # 3.1: Think about types of variables; choose appropriate graphs to find distributions and trends

# Date / Host Growth Over Time  
ggplot(data = airbnb, aes(x = host\_since)) +  
 stat\_ecdf(geom = "step") +  
 ggtitle("AirBnB Growth Over Time (Sydney, Australia)") +  
 xlab("Host Since") +  
 ylab("Percent of Data (n = 10,815)")



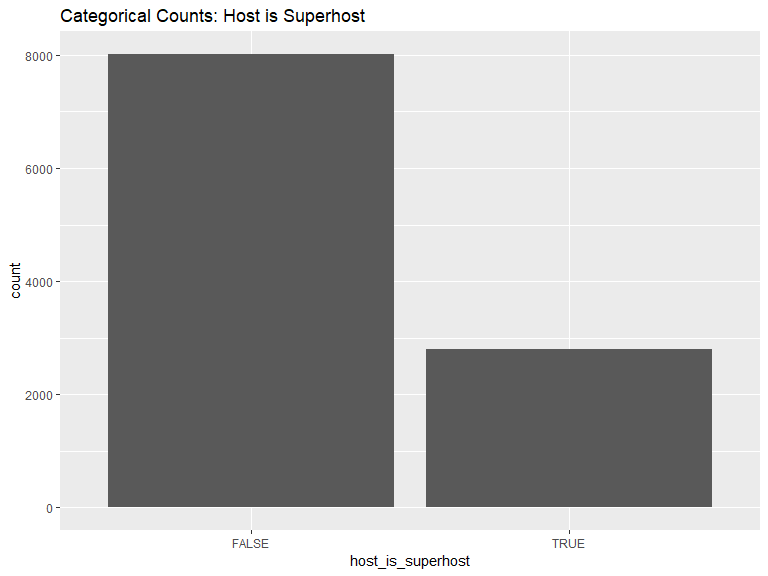
# Categorical Graphs  
  
ggplot(data = airbnb, aes(x = host\_response\_time)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Host Response Time")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



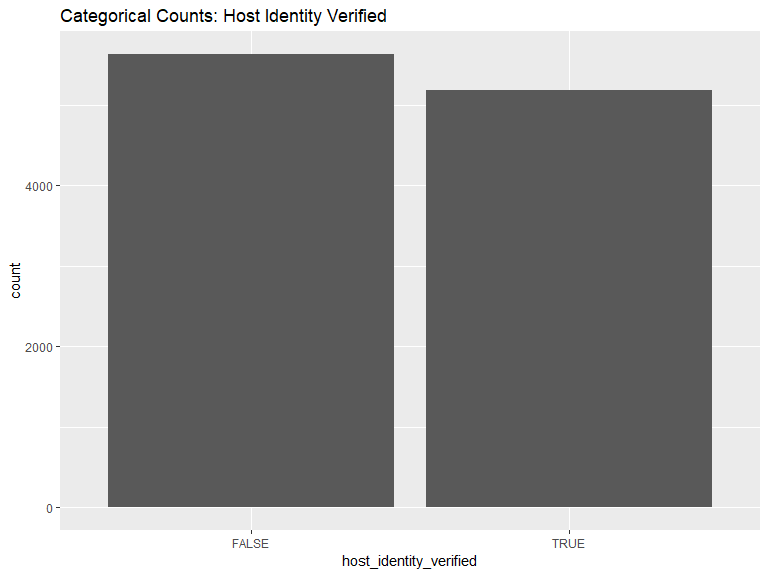
ggplot(data = airbnb, aes(x = host\_is\_superhost)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Host is Superhost")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



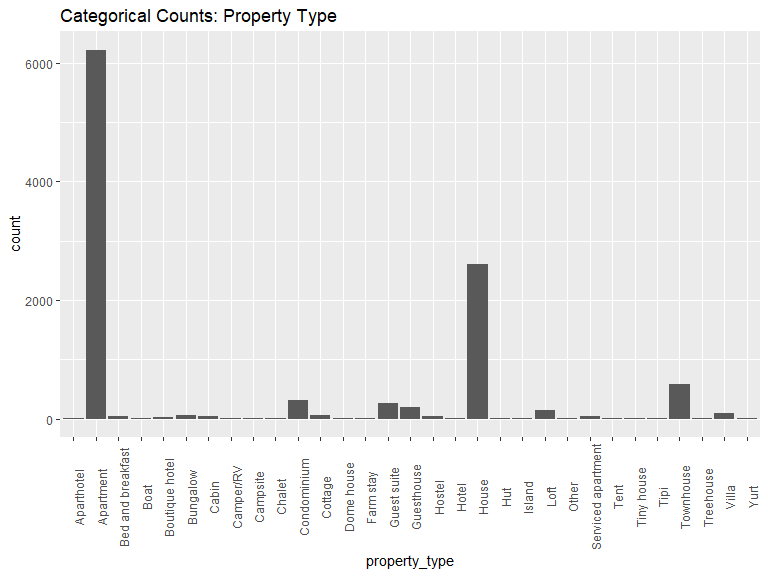
ggplot(data = airbnb, aes(x = host\_identity\_verified)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Host Identity Verified")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



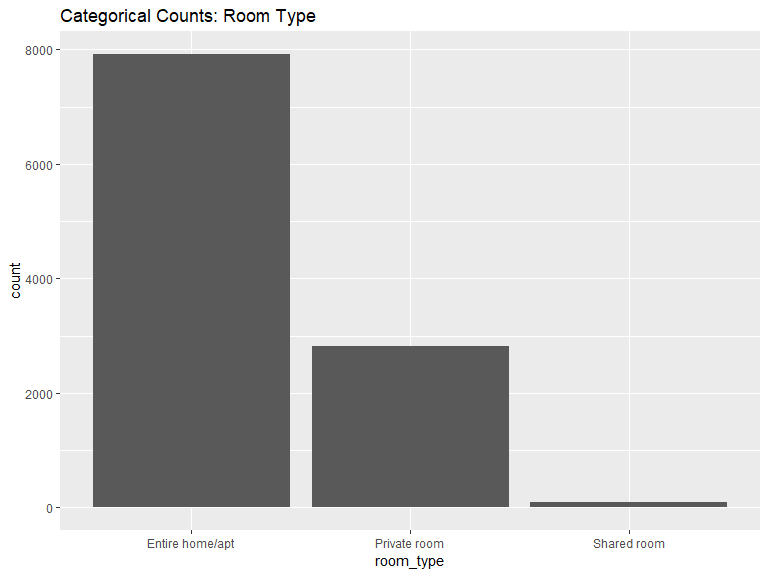
ggplot(data = airbnb, aes(x = property\_type)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Property Type") +  
 theme(axis.text.x = element\_text(angle = 90))

## Warning: Ignoring unknown parameters: binwidth, bins, pad



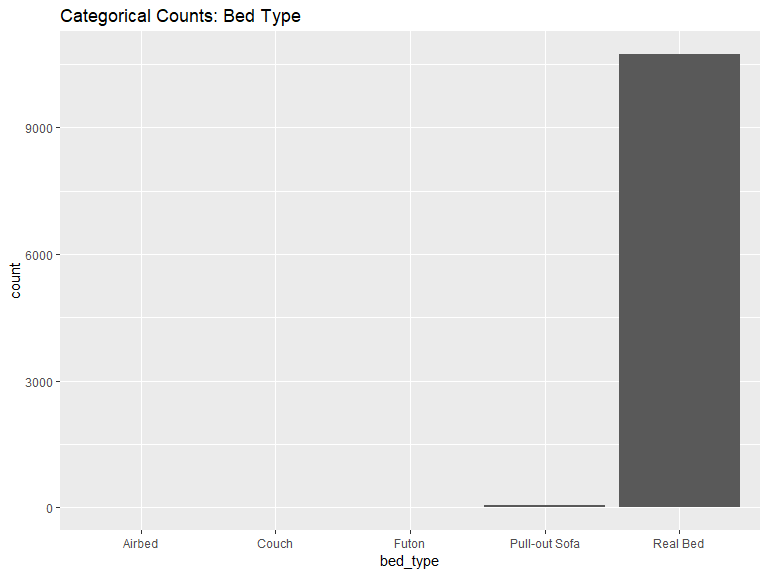
ggplot(data = airbnb, aes(x = room\_type)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Room Type")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



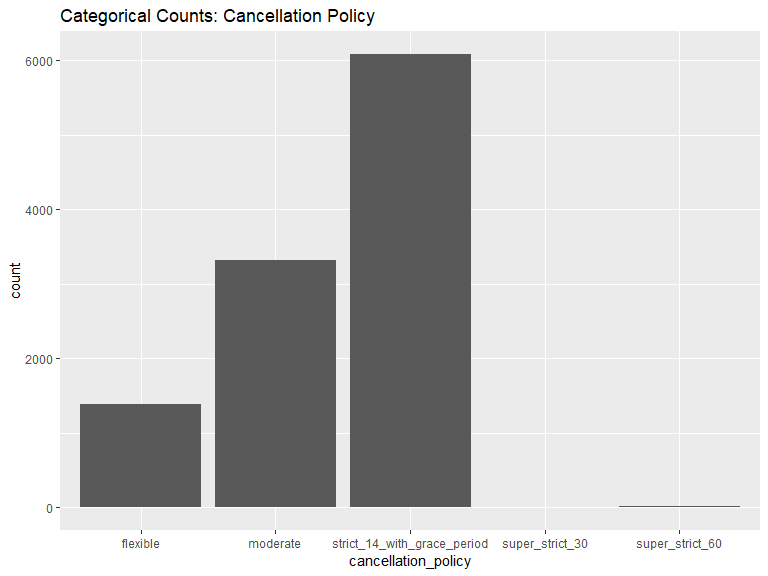
ggplot(data = airbnb, aes(x = bed\_type)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Bed Type")

## Warning: Ignoring unknown parameters: binwidth, bins, pad

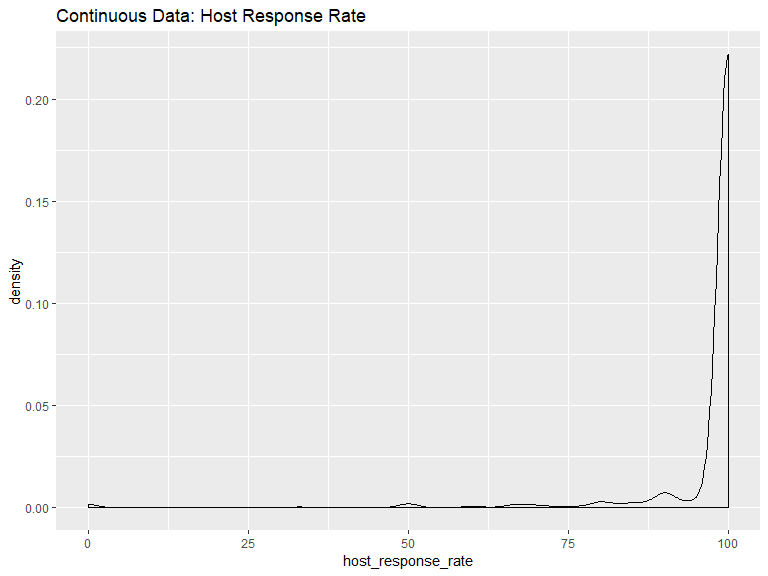


ggplot(data = airbnb, aes(x = cancellation\_policy)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Categorical Counts: Cancellation Policy")

## Warning: Ignoring unknown parameters: binwidth, bins, pad

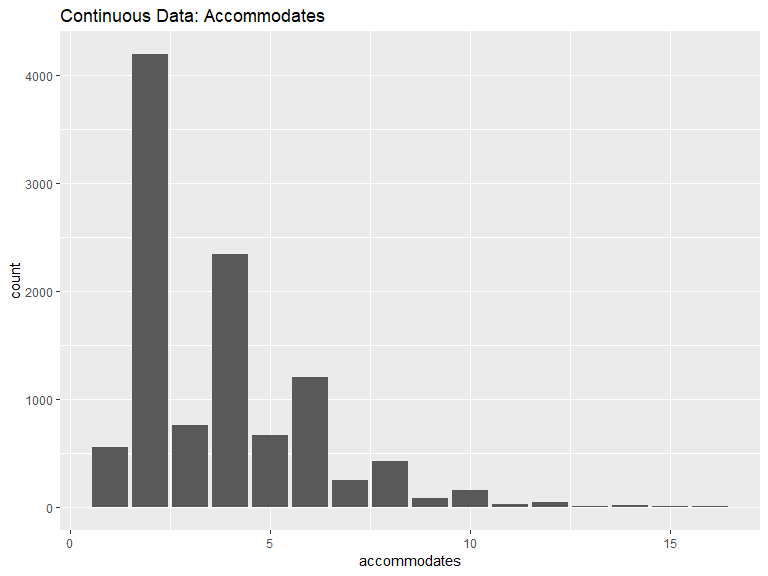


# Continuous Data Graphs  
  
ggplot(data = airbnb, aes(x = host\_response\_rate)) +  
 geom\_density() +  
 ggtitle("Continuous Data: Host Response Rate")



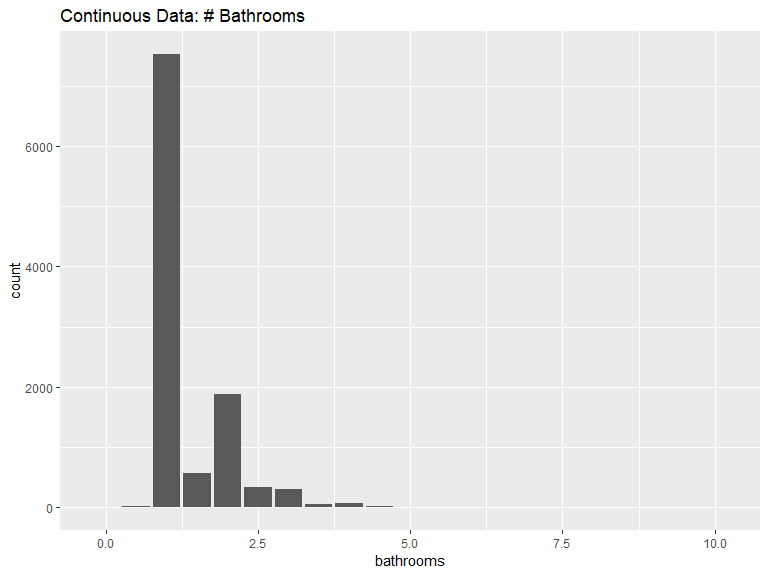
ggplot(data = airbnb, aes(x = accommodates)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: Accommodates")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



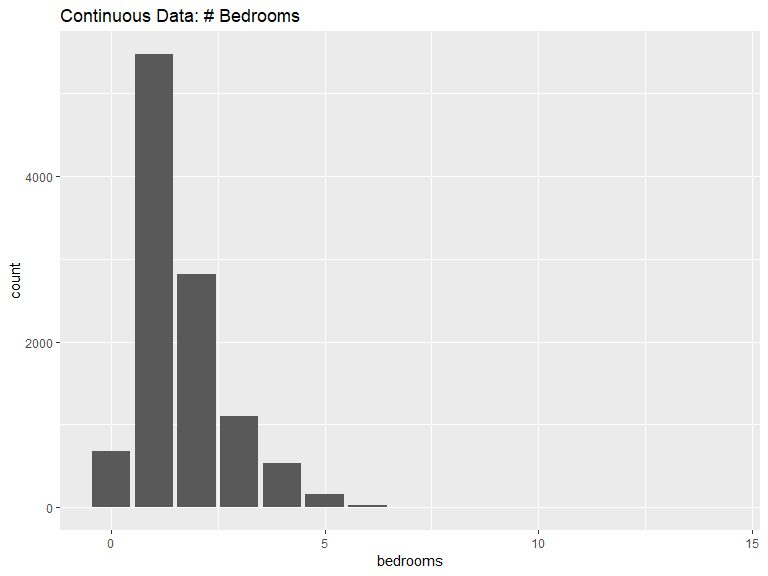
ggplot(data = airbnb, aes(x = bathrooms)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: # Bathrooms")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



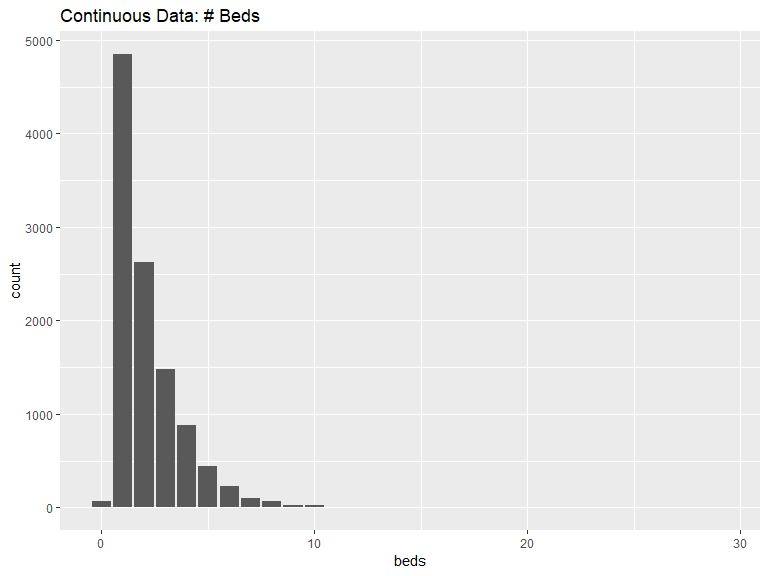
ggplot(data = airbnb, aes(x = bedrooms)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: # Bedrooms")

## Warning: Ignoring unknown parameters: binwidth, bins, pad

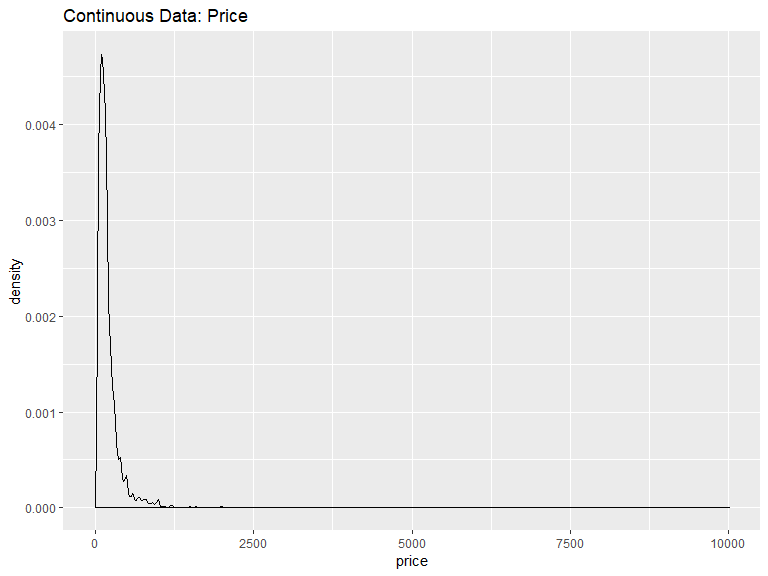


ggplot(data = airbnb, aes(x = beds)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: # Beds")

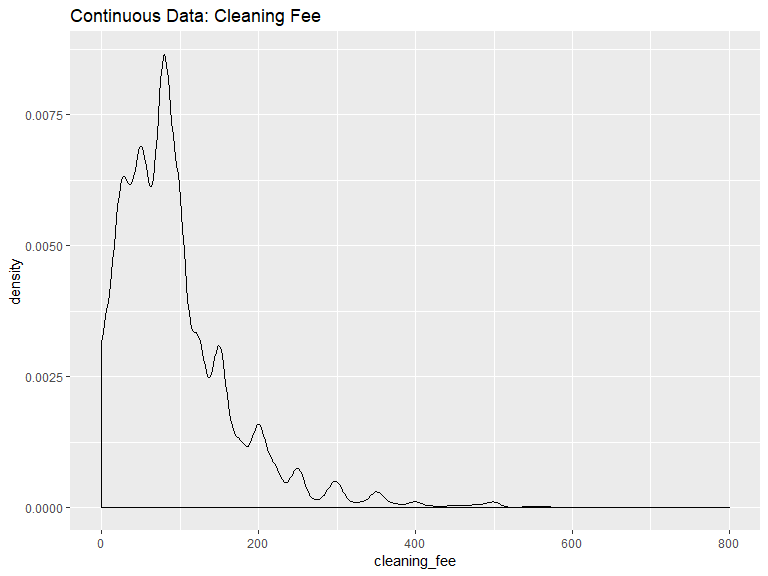
## Warning: Ignoring unknown parameters: binwidth, bins, pad



ggplot(data = airbnb, aes(x = price)) +  
 geom\_density() +  
 ggtitle("Continuous Data: Price")

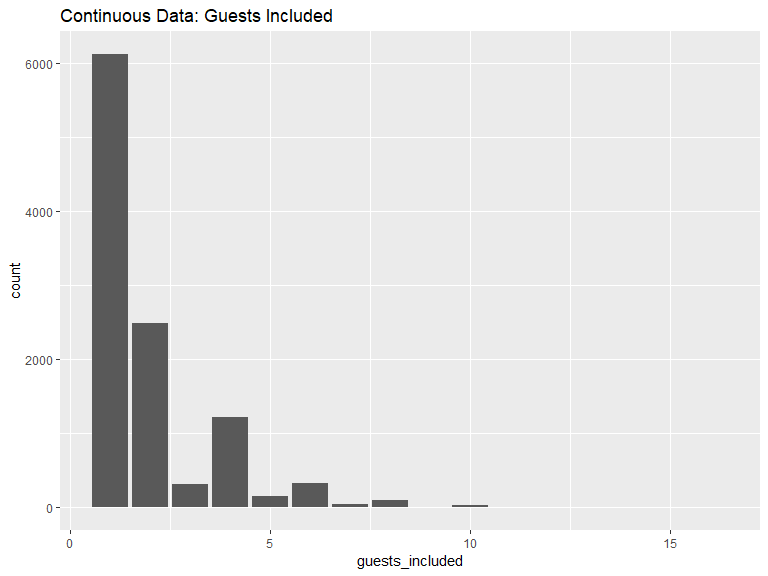


ggplot(data = airbnb, aes(x = cleaning\_fee)) +  
 geom\_density() +  
 ggtitle("Continuous Data: Cleaning Fee")



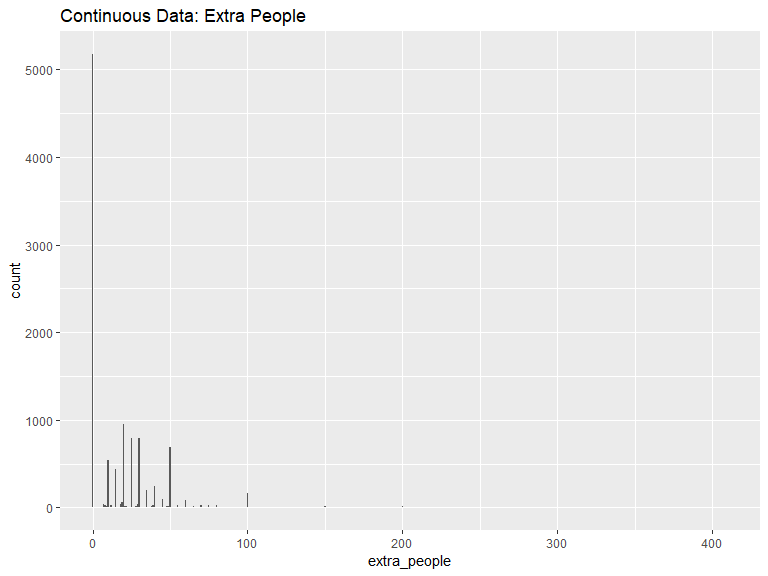
ggplot(data = airbnb, aes(x = guests\_included)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: Guests Included")

## Warning: Ignoring unknown parameters: binwidth, bins, pad



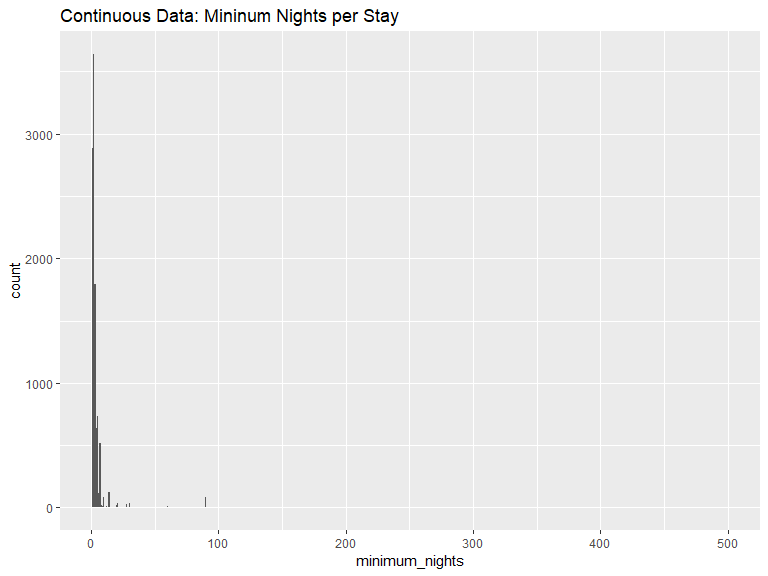
ggplot(data = airbnb, aes(x = extra\_people)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: Extra People")

## Warning: Ignoring unknown parameters: binwidth, bins, pad

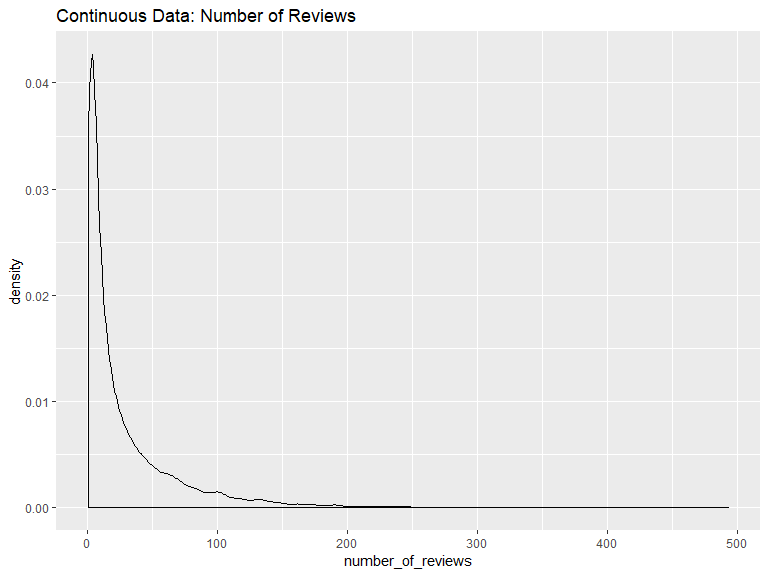


ggplot(data = airbnb, aes(x = minimum\_nights)) +  
 geom\_histogram(stat="count") +  
 ggtitle("Continuous Data: Mininum Nights per Stay")

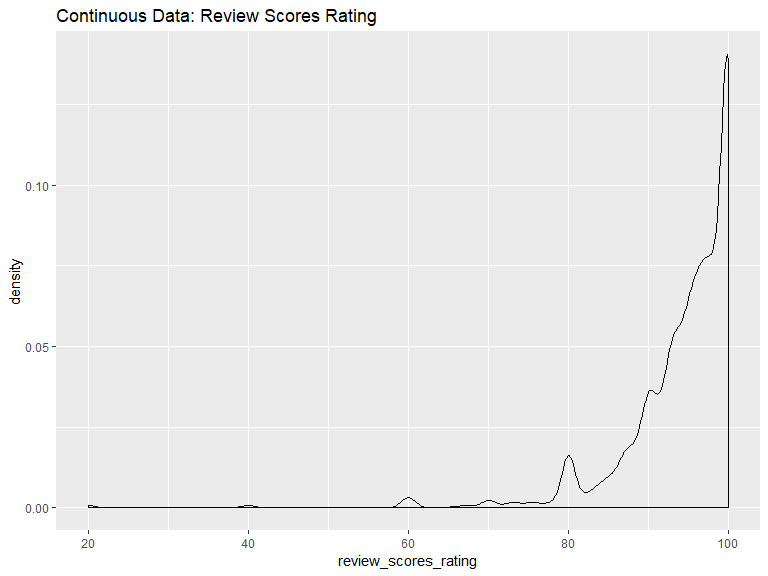
## Warning: Ignoring unknown parameters: binwidth, bins, pad



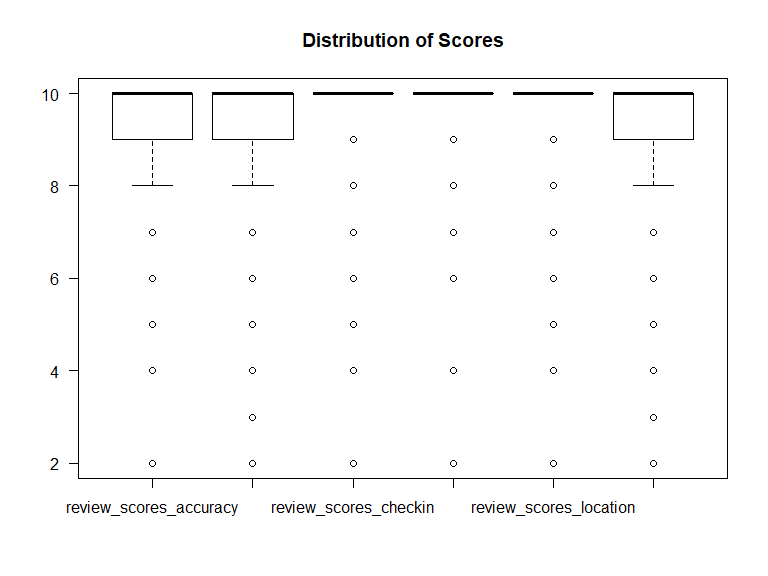
ggplot(data = airbnb, aes(x = number\_of\_reviews)) +  
 geom\_density() +  
 ggtitle("Continuous Data: Number of Reviews")



ggplot(data = airbnb, aes(x = review\_scores\_rating)) +  
 geom\_density() +  
 ggtitle("Continuous Data: Review Scores Rating")



# What is the distribution of the different review scores?  
boxplot(airbnb[29:34], las=1, main = "Distribution of Scores")



# 3.2: Compare different graph types to see which ones best convey trends, outliers, and patterns

For simply display of categorical data, the histogram counts work very well. I omitted the zipcode and city graphs, because there were too many variables to render.

For continuous data, histograms worked well for variables with fewer counts, and density plots worked better for variables with a wider spread of values.

For variables with similar value scale, a boxplot does a good job of showing distribution statistics in a side-by-side fashion.

# 3.3: Describe what you find from the graphs

Some of the categorical values have an even distribution (e.g., host\_verified), but most have values that are more common than others. Bed\_type is mostly “real bed”. Property\_type are predominantly house and apartment, even though there are 31 types.

Most of the review scores are focused on the high end of the scale, so they have a small standard deviations (<1). In retrospect, I could have done a litt

# Q4

# 4.1: Compare and contrast review\_per\_month and number\_of\_reviews

So my general theory is that number of reviews is partly a function of time (listings that have been around longer have more opportunities to be reviewed). So I want to explore what happens if we normalize the number of reviews by the number of months the listing has existed. I thought I could use host\_since, but this is about the *host* and not the *property*.

# Errors of my ways: some host\_ids have multiple listings  
airbnb %>% select(host\_id) %>% table() %>% sort(decreasing = TRUE) %>% head(15)

## .  
## 36410227 15469257 2450066 15651267 7409213 103385102 15193662   
## 156 65 58 53 42 37 35   
## 101139031 21385139 27286333 38478183 33325403 113874 137278159   
## 33 32 30 30 29 28 27   
## 181584188   
## 27

# Affirm that host\_since is the same for all listings, using one of the top host\_ids  
airbnb %>% select(host\_id, host\_since) %>% filter(host\_id == '2450066') %>% head(20)

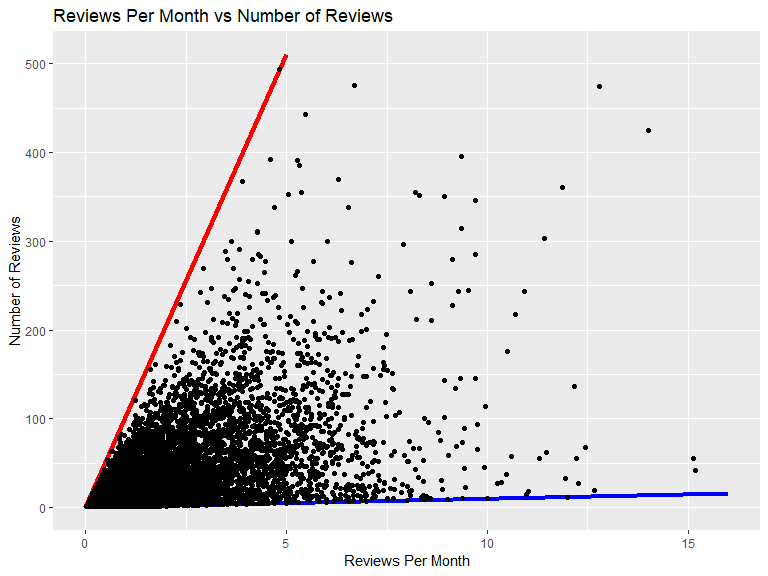
## host\_id host\_since  
## 1 2450066 2012-05-23  
## 2 2450066 2012-05-23  
## 3 2450066 2012-05-23  
## 4 2450066 2012-05-23  
## 5 2450066 2012-05-23  
## 6 2450066 2012-05-23  
## 7 2450066 2012-05-23  
## 8 2450066 2012-05-23  
## 9 2450066 2012-05-23  
## 10 2450066 2012-05-23  
## 11 2450066 2012-05-23  
## 12 2450066 2012-05-23  
## 13 2450066 2012-05-23  
## 14 2450066 2012-05-23  
## 15 2450066 2012-05-23  
## 16 2450066 2012-05-23  
## 17 2450066 2012-05-23  
## 18 2450066 2012-05-23  
## 19 2450066 2012-05-23  
## 20 2450066 2012-05-23

# YUP.  
  
# Find rates  
airbnb <- airbnb %>% mutate(months\_listed = number\_of\_reviews / reviews\_per\_month)  
max(airbnb$months\_listed) # 102 months max; 102.1 \* 5 = 510.5. Use this in the red line.

## [1] 102.0704

Now I have calculated the number of months a listing has been available, but it is calculated using the two variables, so this data (months\_listed) is literally a function of the two. Let’s just plot them and have a look.

# A simple scatterplot of the two variables  
ggplot(airbnb, aes(reviews\_per\_month, number\_of\_reviews)) +  
 ggtitle("Reviews Per Month vs Number of Reviews") +  
 geom\_segment(aes(x = 0,xend = 5, y = 0, yend = 510.5), size = 1.5, color = "red") +  
 geom\_segment(aes(x = 0,xend = 16, y = 0, yend = 16), size = 1.5, color = "blue") +  
 geom\_point() +  
 xlab("Reviews Per Month") +  
 ylab("Number of Reviews")



This graph shows a scatterplot of the two variables against each other. The blue line represents the max rate (1:1, that is, 30 reviews in one month). So a point near the blue line is receiving reviews as frequently as possible. The red line represents the minimum number of reviews a property could receive, for the given reviews per month and the maximum age of a listing in this dataset (102.1 months).

Now let’s look at overlap between the top 100 of each variable

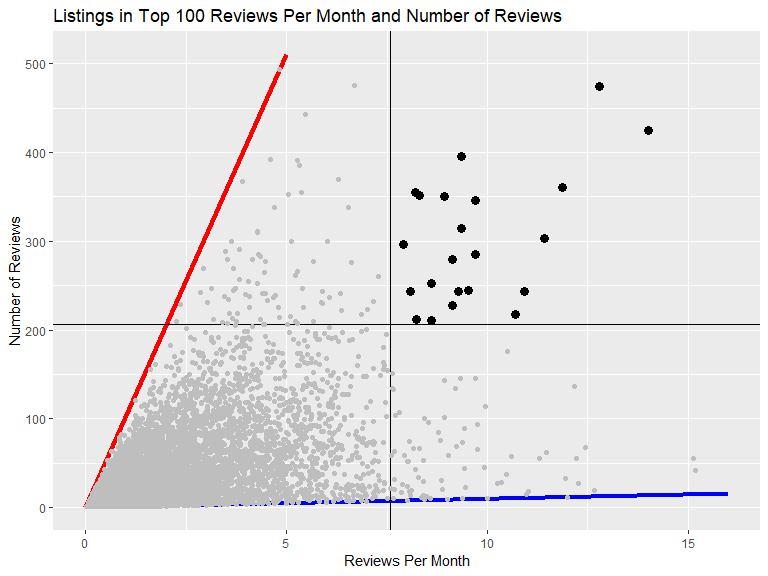
#head(sort(airbnb$reviews\_per\_month, decreasing = TRUE), 100) # DELETE  
# order() pulls the row values; I can use these to subset the IDs  
top100rpm <- airbnb$id[head(order(airbnb$reviews\_per\_month, decreasing = TRUE), 100)]  
  
#head(sort(airbnb$number\_of\_reviews, decreasing = TRUE), 100)  
top100nrev <- airbnb$id[head(order(airbnb$number\_of\_reviews, decreasing = TRUE), 100)]  
  
# How many values in the top 100 rate (rev per mo) are in the top 100 total (num of reviews)  
length(intersect(top100rpm, top100nrev))

## [1] 22

# Which IDs are in both lists?  
intersect(top100rpm, top100nrev)

## [1] 12954762 7944819 13279754 15257722 15424010 17946780 10111445  
## [8] 13623082 7935975 5751561 11589811 15186470 15685660 13193475  
## [15] 8412341 13582232 16123508 5796898 15474118 6327407 13499531  
## [22] 9352316

ggplot(airbnb, aes(reviews\_per\_month, number\_of\_reviews)) +  
 ggtitle("Listings in Top 100 Reviews Per Month and Number of Reviews") +  
 geom\_segment(aes(x = 0,xend = 5, y = 0, yend = 510.5), size = 1.5, color = "red") +  
 geom\_segment(aes(x = 0,xend = 16, y = 0, yend = 16), size = 1.5, color = "blue") +  
 geom\_vline(xintercept = min(airbnb$reviews\_per\_month[airbnb$id %in% top100rpm])) +  
 geom\_hline(yintercept = min(airbnb$number\_of\_reviews[airbnb$id %in% top100nrev])) +  
 geom\_point(color = "grey") +  
 geom\_point(data = airbnb[airbnb$id %in% intersect(top100rpm, top100nrev),], size = 3) +  
 xlab("Reviews Per Month") +  
 ylab("Number of Reviews")



## Discussion and findings

These two variables (number of reviews and reviews per month) both describe the populariy of a listing, but in different ways. Number of reviews is the total number of reviews. Properties with a high value here will have had the most guests visit. However, this metric is influenced by the length of time it has been around. Reviews per month attempts to account for the time a listing has been offered by normalizing the data against the number of months is has been available (and earlier I back-calculated the number of months). New listings with a strong opening can lead this metric. One potential drawback to the reviews\_per\_month variable is that it penalizes listings with higher minimum night stays. Listings with a seven-night minimum would have a four review\_per\_month ceiling.

22 listings are in both the top 100 review\_per\_month and the top 100 number\_of\_reviews. I have added reference lines to the graph illustrating the minimum value in the top 100 of each variable, then highlighted the values in the top 100 of each (the upper right quadrant).

The upper-right quadrant is the **best of the best** – popular for a long time and frequently reviewed. The upper-left quadrant contains **established favorites**. These properties have a lower review frequency, but they have been around long enough to achieve high review counts. The lower-right quadrant contains the **best new listings**. These listings have low counts, but are being reviewed frequently enough that it’s expected they will work their way into the higher property counts. The lower-left quadrant contains **typical listings**.

Ultimately, the right quadrants can be summarized as popular for a long time (upper right) and popular for a short time (lower right). Thus, reviews\_per\_month is better suited as a measure of popularity.

# 4.2: Analyze at least three other groups as in 4.1

# 4.2.1: Cleaning Fee vs Price

Do expensive listings have higher cleaning fees? Or are hosts making up for low prices with high cleaning fees?

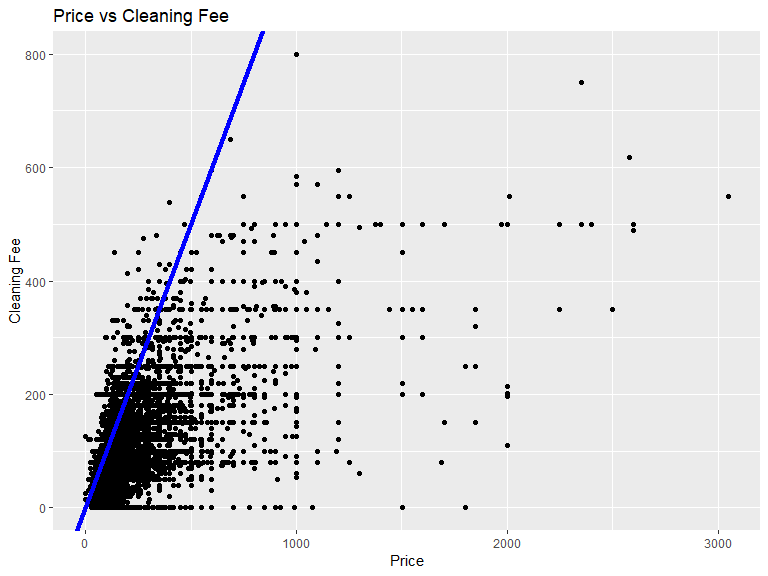
airbnb$cleaning\_fee\_pct <- airbnb$cleaning\_fee / airbnb$price  
summary(airbnb$cleaning\_fee\_pct) # bad news, we have infinite values

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.3077 0.4762 Inf 0.7080 Inf

airbnb %>% filter(price == 0) # dumb news, people have airbnbs with a price of 0

## id  
## 1 20563580  
## 2 20718560  
## 3 21372128  
##

# Some outlier expensive price listings were messing up the graph, so I filtered them  
ggplot(airbnb %>% filter(price < 5000), aes(price, cleaning\_fee)) +  
 ggtitle("Price vs Cleaning Fee") +  
 geom\_point() +  
 geom\_abline(slope = 1, color = "blue", size = 1.5) +  
 xlab("Price") +  
 ylab("Cleaning Fee")



# 953 listings have a cleaning fee higher than the price  
airbnb %>% filter(cleaning\_fee > price) %>% count()

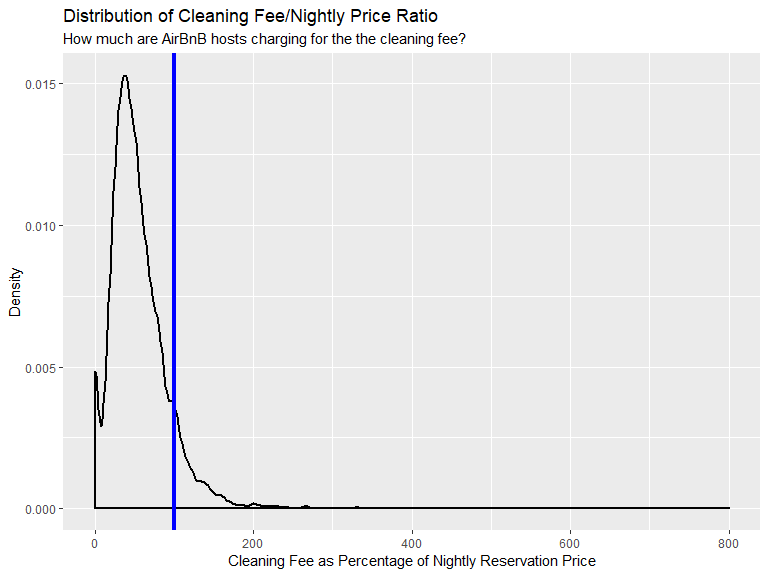
## # A tibble: 1 x 1  
## n  
## <int>  
## 1 953

# This is 8.81% of all listings!  
airbnb %>% filter(cleaning\_fee > price) %>% count() / airbnb %>% count() \* 100

## n  
## 1 8.811835

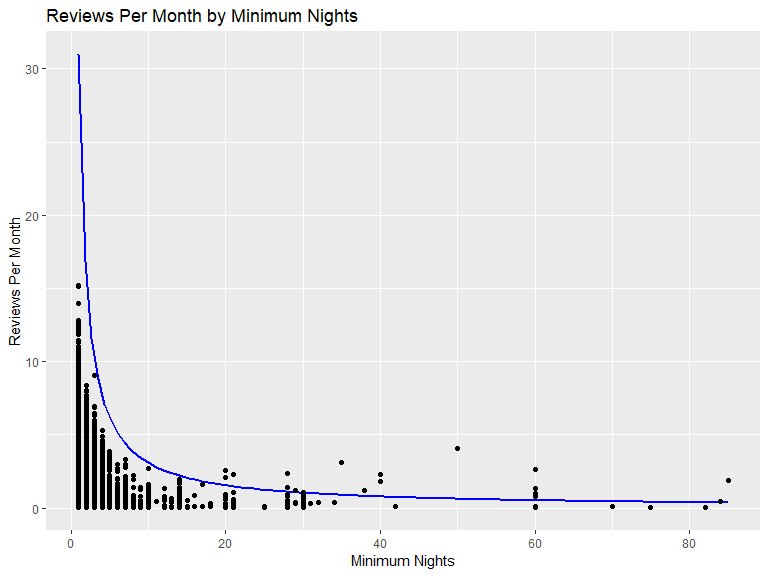
# What does the distribution of cleaning fee percentages look like?  
ggplot(data = airbnb, aes(x = cleaning\_fee\_pct \* 100)) +  
 geom\_density(size = 1) +  
 ggtitle("Distribution of Cleaning Fee/Nightly Price Ratio", subtitle = "How much are AirBnB hosts charging for the the cleaning fee?") +  
 geom\_vline(xintercept = 100, color = "blue", size = 1.5) +  
 xlab("Cleaning Fee as Percentage of Nightly Reservation Price") +  
 ylab("Density")

## Warning: Removed 3 rows containing non-finite values (stat\_density).

 # DISCUSS # PHIL - do not forget that you imputed a number of thPHILe median values ($80) # Consider Minimum Night Stays \* Price, calc cleaning fee as pct of THAT

# 4.2.2: Minimum Night Stays

ggplot(data = airbnb %>% filter(minimum\_nights < 90), aes(minimum\_nights, reviews\_per\_month)) +  
 stat\_function(fun = function(x) 31/x, color = "blue", size = 1) +  
 geom\_point() +  
 ggtitle("Reviews Per Month by Minimum Nights") +  
 xlab("Minimum Nights") +  
 ylab("Reviews Per Month")

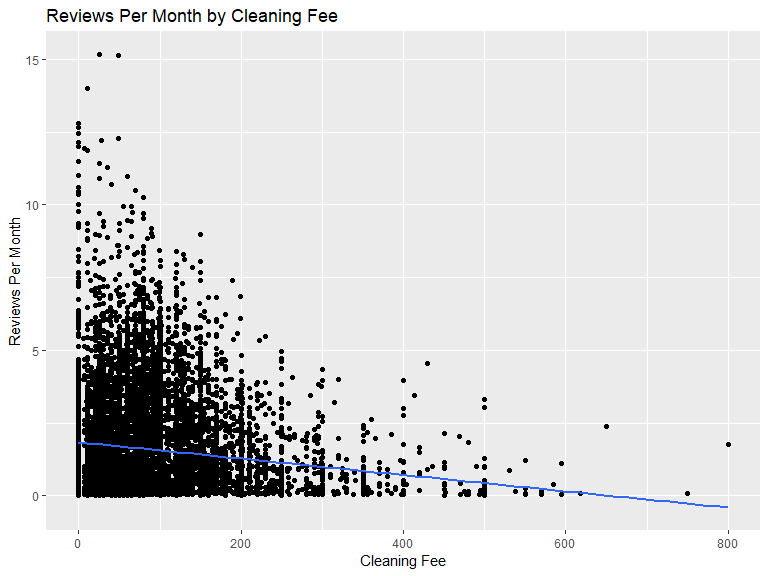


Something is wrong here. How can a listing with a 60-day minimum have 2 reviews per month? I filtered out minimum stays of 90 or more, because they are outliers and didn’t add useful information. I have added a blue curve, which is the theoretical maximum reviews\_per\_month a listing can have (assumes all stays are for minimum length and 0% vacancy, and a 31-day month). It is possible that the listing has increased the minimum stay after amassing a high number/frequency of reviews.

# 4.2.3: Reviews\_Per\_Month

Because I determined reviews\_per\_month was a good indicator of popularity, I thought I would graph some other variables against it in a scatterplot and see if any patterns arise.

ggplot(data = airbnb, aes(cleaning\_fee, reviews\_per\_month)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se=FALSE) +  
 ggtitle("Reviews Per Month by Cleaning Fee") +  
 xlab("Cleaning Fee") +  
 ylab("Reviews Per Month")



# Findings

Here I have a graph showing how the reviews\_per\_month relate to the cleaning fee. The blue line is the line of best fit. With a negative slope, this implies that as the cleaning fee increases, the reviews\_per\_month decreases.

# Q5

**Propose three different hypotheses for business analysis**

As a hypothetical AirBnB host, I’m looking at three ways in which to upgrade my listing. My goal is to either improve my overall rating from guests or be able to raise the price – or potentially both. The three upgrades I am considering are: a Pool, Cable TV subscription, or a BBQ Grill.

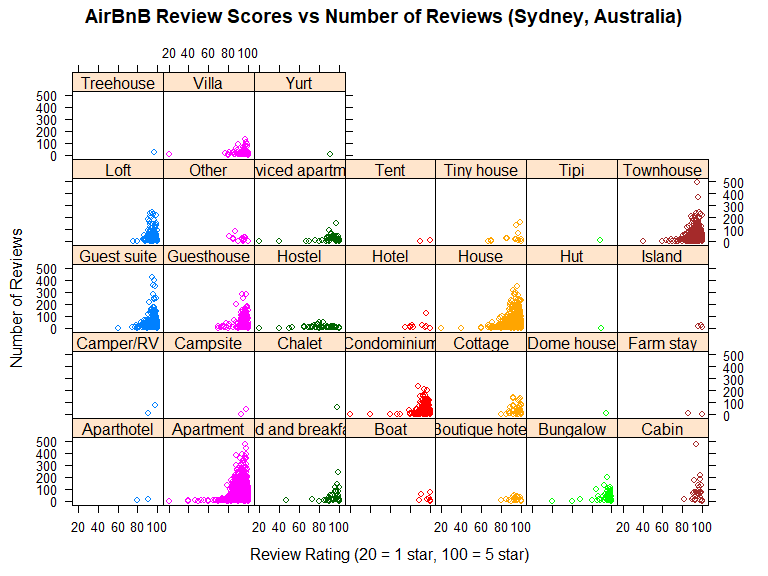
* Listings with a Pool will have a higher price than those without
* Listings with Cable TV will have a higher price than those without
* Listings with a BBQ Grill will have a higher price than those without

PART 2: Data Analysis

# Q6

**6.1: Make ONE plot to visualize relationship between review\_scores\_rating and number of reviews for all categories of property\_type. Explain your findings.**

xyplot(airbnb$number\_of\_reviews ~ airbnb$review\_scores\_rating | airbnb$property\_type, groups = airbnb$property\_type, xlab = "Review Rating (20 = 1 star, 100 = 5 star)", ylab = "Number of Reviews", main="AirBnB Review Scores vs Number of Reviews (Sydney, Australia)")



# Findings

Although certain property types are more common than others, the distributions are largely similar: it is rare for a rental to get over 100 reviews if it is not at least a four-star property (rating = 80). This is why under the 100 review line, a variety of scores can be found (although still concentrated toward four- and five-star reviews). Poorly-reviewed rentals will see fewer guests and thus, reviews.

I’m unsure if all of these property types have been available for the same length of time. For example, hostels and boutique hotels seem to have a moderate number of reviews, but none over 100. I wonder if this is a newer addition to the property types.

# 6.2: Make ONE plot to show relationship among property types, room types, bed types, and reviews per month. Explain your findings.

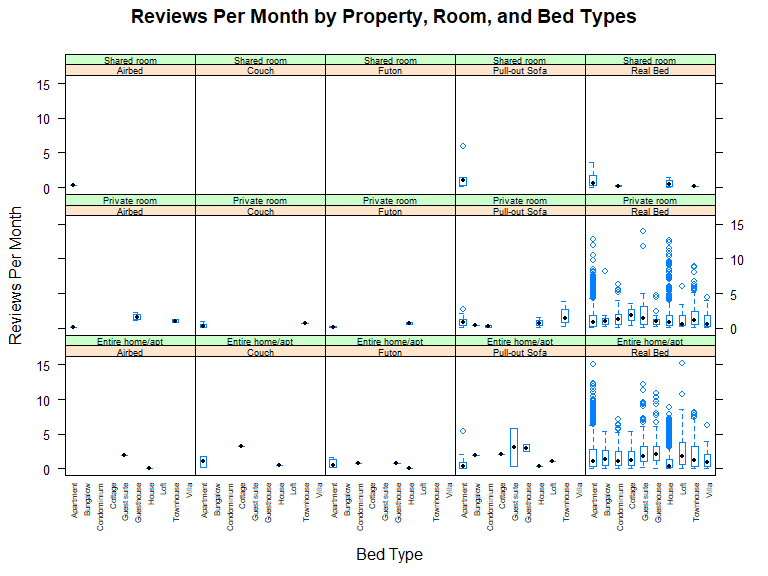
# Get a target row count (10,546)  
sum(head(sort(table(airbnb$property\_type), decreasing = TRUE), 10))

## [1] 10546

# Subset the data frame to top property types  
# Get a vector of property types  
i <- names(head(sort(table(airbnb$property\_type), decreasing = TRUE), 10))  
# Subset  
prairbnb <- airbnb[airbnb$property\_type %in% i,]  
# A double check  
table(prairbnb$property\_type)

##   
## Apartment Bungalow Condominium Cottage Guest suite Guesthouse   
## 6222 62 309 59 258 200   
## House Loft Townhouse Villa   
## 2604 150 589 93

# For best results, view on an 80" plasma 4k TV  
bwplot(prairbnb$reviews\_per\_month ~ prairbnb$property\_type | prairbnb$bed\_type + prairbnb$room\_type,  
 scales=list(x=list(rot=90, cex=0.5)),  
 xlab='Bed Type', ylab='Reviews Per Month',  
 par.strip.text=list(cex=.55),  
 cex = .5,  
 main = "Reviews Per Month by Property, Room, and Bed Types")

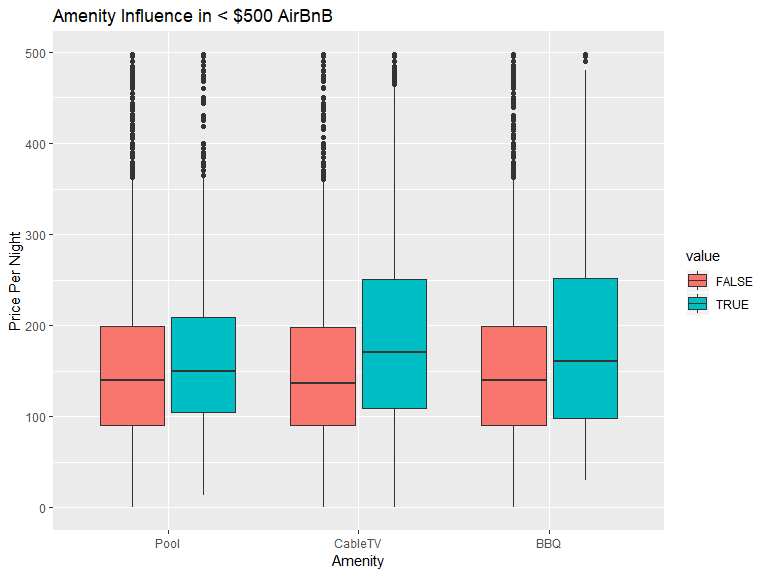


# Findings

This layout is a grid of plots: each row contains one of three room types, and each column contains one of five bedtypes. The property types are shown in the boxplot inside each grid cell. This layout shows that “real bed” is the most common bed type (due to the visual density of the boxplots). Real beds are also found at the most popular listings: all listings with over 10 reviews per month have a real bed. Futons, couches, and airbeds are not common, and the listings that have them do not have more than five reviews per month. The pull-out sofa is the second-most popular bed type, and it has a couple listings that break the five review per month level. In the Real Bed > Entire Home and Private Room graphs, the large amount of outliers above the whiskers indicate a positively-skewed distribution.

# 6.3: Make some plots to explore hypotheses in Q5. Explain your choice and describe interesting findings.

# Create the variables I need. grep() returns row numbers (i.e. row names)  
airbnb$has\_pool <- rownames(airbnb) %in% grep("Pool", airbnb$amenities)  
airbnb$has\_cable <- rownames(airbnb) %in% grep("Cable TV", airbnb$amenities)  
airbnb$has\_bbq <- rownames(airbnb) %in% grep("BBQ ", airbnb$amenities)  
  
# ggplot boxplots require data in long format. melt() in reshape2 can help with that  
mair <- airbnb %>% filter(price < 500) %>% select(c("id", Pool = "has\_pool", CableTV = "has\_cable", BBQ = "has\_bbq", "price")) %>% melt(id=c("id", "price"))  
ggplot(mair, aes(x=factor(variable), y=price)) +   
 geom\_boxplot(aes(fill = value))+  
 ggtitle("Amenity Influence in < $500 AirBnB") +  
 xlab("Amenity") +  
 ylab("Price Per Night")



# Get the numbers  
mair %>% group\_by(variable, value) %>% summarise(count = n(), mean\_price = mean(price))

## # A tibble: 6 x 4  
## # Groups: variable [?]  
## variable value count mean\_price  
## <fct> <lgl> <int> <dbl>  
## 1 Pool FALSE 8451 161.  
## 2 Pool TRUE 1748 173.  
## 3 CableTV FALSE 8113 155.  
## 4 CableTV TRUE 2086 195.  
## 5 BBQ FALSE 8645 159.  
## 6 BBQ TRUE 1554 189.

First, I filtered the data to eliminate any properties over $500/night. I did this because my hypothetical rental is not in the extreme luxury class. I want to analyze my peer class. Second, I had to format this data into a long table (due to ggplot’s handling of boxplot). This has the benefit of placing my plots next to each other for easy comparison.

The boxplot shows that the presensce of all of my potential amenities increase the price (at mean and at IQR points). The table indicates that pool, cable, and bbq will increase the mean by 12, 40, and 30 dollars per night. To my surprise, the pool had the smallest effect on price, and was also associated with the lowest prices. Perhaps a pool is less novel in Sydney? Due to the large cost of installation, ongoing costs, and small return, I will rule out the pool.

Cable TV and BBQ Grill are both cost-effective. Cable has an ongoing cost that will eat into profits, so I will compare the monthly cost against my occupancy rates and my planned price increase.

The BBQ Grill is more of a one-time expense with a strong return. I will choose that one first.

# Q7

# 7.1: Clean the price

Price was cleaned back in step 1 with the following:

# airbnb$price <- as.numeric(gsub("^\\$|,","",airbnb$price))

# 7.2: Add number of amenities as column

# Amenities are separated by a comma and opened with a curly brace. Count the curly brace and commas for num of amenities  
# This statement is using lapply to make a vector 10815 elements long. gregexpr returns a list, and I need to get the   
# length of the first element of the list  
airbnb$number\_of\_amenities <- sapply(airbnb$amenities, function(x) length(gregexpr("\\{|,",x)[[1]]))  
  
# Problem with the above is that it counts empty curly brace as 1  
airbnb$number\_of\_amenities <- sapply(airbnb$amenities, function(x) length(strsplit(x,",")[[1]]))  
# This does the same.   
  
# Clean up the ones; using a within() statment to minimize typing airbnb$  
airbnb <- within(airbnb, number\_of\_amenities[number\_of\_amenities == 1] <- 0)

# 7.3: Calculate mean review\_scores\_rating against cancellation policies. What do you find?

# Using Base R  
by(airbnb$review\_scores\_rating, airbnb$cancellation\_policy, mean)

## airbnb$cancellation\_policy: flexible  
## [1] 94.15888  
## --------------------------------------------------------   
## airbnb$cancellation\_policy: moderate  
## [1] 95.00604  
## --------------------------------------------------------   
## airbnb$cancellation\_policy: strict\_14\_with\_grace\_period  
## [1] 93.77139  
## --------------------------------------------------------   
## airbnb$cancellation\_policy: super\_strict\_30  
## [1] 80  
## --------------------------------------------------------   
## airbnb$cancellation\_policy: super\_strict\_60  
## [1] 89.8

# Using dplyr  
airbnb %>% group\_by(cancellation\_policy) %>% summarise(mean = mean(review\_scores\_rating))

## # A tibble: 5 x 2  
## cancellation\_policy mean  
## <chr> <dbl>  
## 1 flexible 94.2  
## 2 moderate 95.0  
## 3 strict\_14\_with\_grace\_period 93.8  
## 4 super\_strict\_30 80   
## 5 super\_strict\_60 89.8

# Findings

The super-strict cancellation policies have the worst scores – under 90. These are less guest-friendly. The highest mean was for the moderate policy, which sits between flexible and strict\_14\_with\_grace\_period on the guest-friendly scale. Why wouldn’t flexible – as the most guest-friendly – have the highest ratings? Perhaps the hosts who choose the flexible policy are more care-free and less professional in their rental? It could also be statistical noise; are 95.00 and 94.15 meaningfully different?

# Here is a t-test  
t.test(airbnb$review\_scores\_rating[airbnb$cancellation\_policy == "flexible"],airbnb$review\_scores\_rating[airbnb$cancellation\_policy == "moderate"])

##   
## Welch Two Sample t-test  
##   
## data: airbnb$review\_scores\_rating[airbnb$cancellation\_policy == "flexible"] and airbnb$review\_scores\_rating[airbnb$cancellation\_policy == "moderate"]  
## t = -3.2849, df = 2062.9, p-value = 0.001037  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -1.3529094 -0.3414036  
## sample estimates:  
## mean of x mean of y   
## 94.15888 95.00604

# I have run the t-test, and I THINK that because the p value is < .05, I will reject the null hypothesis.  
# Then again, I feel in this class we would find the standard deviation which for "flexible" is 8.67.  
sd(airbnb$review\_scores\_rating[airbnb$cancellation\_policy == "flexible"])

## [1] 8.665896

# In this case, I could argue that the mean for "strict\_14" falls within +/- 8.67 of the mean flexible. Therefore they are  
# not statistically different.

# Other manipulations

Here I use dplyr to get mean scores by types

airbnb %>% group\_by(property\_type) %>% summarise(mean = mean(review\_scores\_rating)) %>% arrange(desc(mean))

## # A tibble: 31 x 2  
## property\_type mean  
## <chr> <dbl>  
## 1 Island 98.3  
## 2 Chalet 98   
## 3 Boat 97   
## 4 Treehouse 97   
## 5 Cottage 95.9  
## 6 Cabin 95.9  
## 7 Loft 95.6  
## 8 Guest suite 95.5  
## 9 Campsite 95.5  
## 10 Farm stay 95.3  
## # ... with 21 more rows

airbnb %>% group\_by(room\_type) %>% summarise(mean = mean(review\_scores\_rating)) %>% arrange(desc(mean))

## # A tibble: 3 x 2  
## room\_type mean  
## <chr> <dbl>  
## 1 Private room 94.6  
## 2 Entire home/apt 94.1  
## 3 Shared room 84.9

airbnb %>% group\_by(bed\_type) %>% summarise(mean = mean(review\_scores\_rating)) %>% arrange(desc(mean))

## # A tibble: 5 x 2  
## bed\_type mean  
## <chr> <dbl>  
## 1 Futon 94.6  
## 2 Real Bed 94.2  
## 3 Pull-out Sofa 92.6  
## 4 Couch 92   
## 5 Airbed 81.8

airbnb %>% group\_by(number\_of\_amenities) %>% summarise(mean = mean(review\_scores\_rating)) %>% arrange(desc(mean))

## # A tibble: 87 x 2  
## number\_of\_amenities mean  
## <dbl> <dbl>  
## 1 80 100   
## 2 95 100   
## 3 97 100   
## 4 79 99   
## 5 81 99   
## 6 69 98.6  
## 7 77 98.5  
## 8 89 98   
## 9 101 98   
## 10 74 97.6  
## # ... with 77 more rows

I added amenity data in 6.3. I will add word count data in 10. I will add spatial data in part 4.

# Q8: Linear Modeling

**Explain 10 variables; evaluate one**

In question four, I established that reviews\_per\_month was the preferred measure of popularity (the right quadrants were either popular for a long time, or popular for a short time). I will use reviews\_per\_month as the dependent variable in this exercise.

Reviews per month is a mix of occupancy rate and minimum night stay.

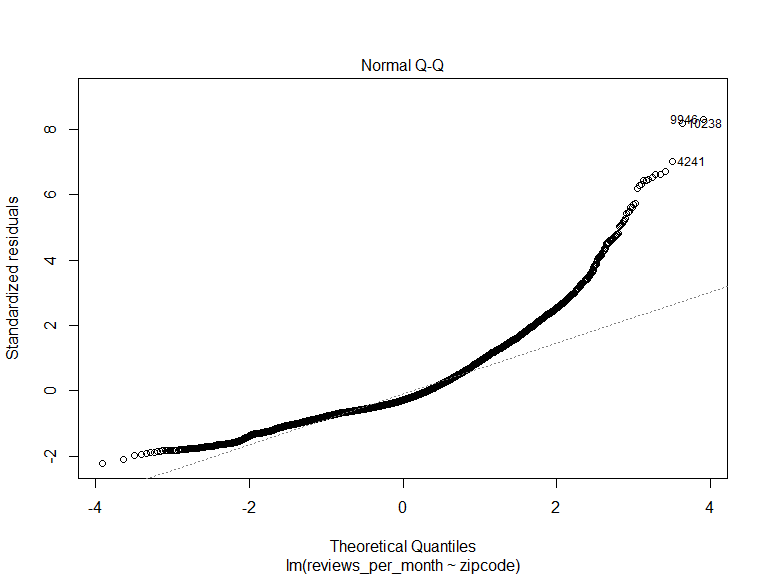
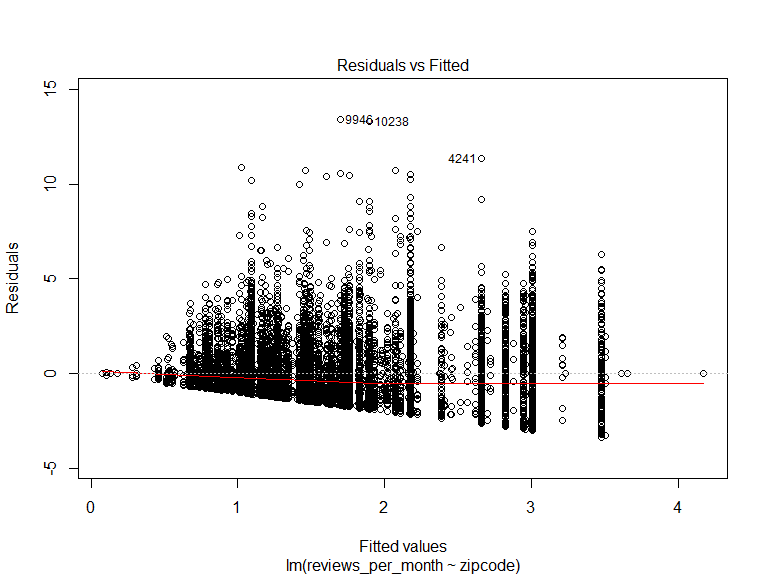
* Price: I don’t think price will affect reviews\_per month. Expensive listings could be frequently reviewed (R^2 = 0.008)
* Host\_is\_superhost: (R^2 = 0.088)
* Number\_of\_verifications: This could help, but I’m not sure that that buyers care more about the NUMBER of verifications, but just that there is at least one.
* Cleaning\_fee: This might have a small impact (R^2 = 0.01566)
* Review\_scores\_communication: Perhaps hosts with good communication skills will be more likely to receive reviews? Apparently not (R^2 = 0.006)
* Host\_response\_rate: Building off the previous variable, perhaps responsive hosts will be more likely to receive reviews? (R^2 = 0.009)
* Room\_type: Perhaps shared\_room will be less popular than a private room, which may be less popular than an entire house/apt? (R^2 = 0.005)
* Number\_of\_amentities: I think guests will be attracted to listings with a number of amenities, because 1. guests like amenities and 2. multiple amenities can be found on listings with complete descriptions. (R^2 = 0.009)
* Zipcode: I believe certain zipcodes will be more popular than others. (R^2 = 0.147)
* Minimum\_nights: I think this one will have the most correlation (negative slope). (R^2 = 0.015)

# Zipcode was my winning -- although weak -- correlation. Here's the model and charts  
amodel <- lm(reviews\_per\_month ~ zipcode, data = airbnb)  
summary(amodel)

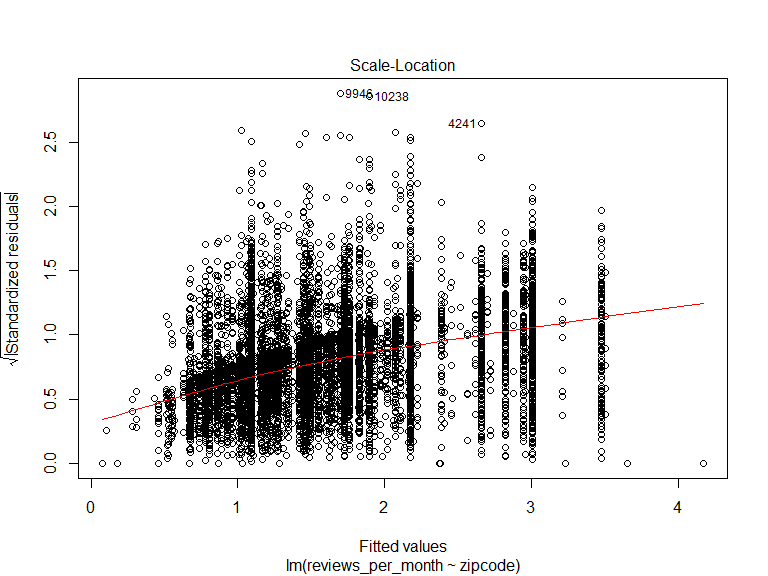
##   
## Call:  
## lm(formula = reviews\_per\_month ~ zipcode, data = airbnb)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.3773 -1.0018 -0.4533 0.6927 13.4386   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.00178 0.06932 43.304 < 2e-16 \*\*\*  
## zipcode2007 -0.05871 0.16016 -0.367 0.713960   
## zipcode2008 -0.34467 0.13771 -2.503 0.012333 \*   
## zipcode2009 -0.17768 0.13883 -1.280 0.200614   
## zipcode2010 -0.82550 0.08847 -9.331 < 2e-16 \*\*\*  
## zipcode2011 -1.27714 0.10390 -12.292 < 2e-16 \*\*\*  
## zipcode2015 -1.30041 0.17539 -7.414 1.31e-13 \*\*\*  
## zipcode2016 -1.17478 0.13196 -8.903 < 2e-16 \*\*\*  
## zipcode2017 -1.10717 0.12813 -8.641 < 2e-16 \*\*\*  
## zipcode2018 -1.40035 0.20646 -6.783 1.24e-11 \*\*\*  
## zipcode2019 -1.50821 0.44036 -3.425 0.000617 \*\*\*  
## zipcode2020 -0.93095 0.17995 -5.173 2.34e-07 \*\*\*  
## zipcode2021 -1.55448 0.13716 -11.333 < 2e-16 \*\*\*  
## zipcode2022 -1.92912 0.14308 -13.483 < 2e-16 \*\*\*  
## zipcode2023 -2.32395 0.20779 -11.184 < 2e-16 \*\*\*  
~~~ manually truncated by author  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.627 on 10609 degrees of freedom  
## Multiple R-squared: 0.1468, Adjusted R-squared: 0.1304   
## F-statistic: 8.907 on 205 and 10609 DF, p-value: < 2.2e-16

plot(amodel, ask = FALSE)

## Warning: not plotting observations with leverage one:  
## 1826, 2608, 6910, 7042, 7195, 7661, 7719, 8222, 9759

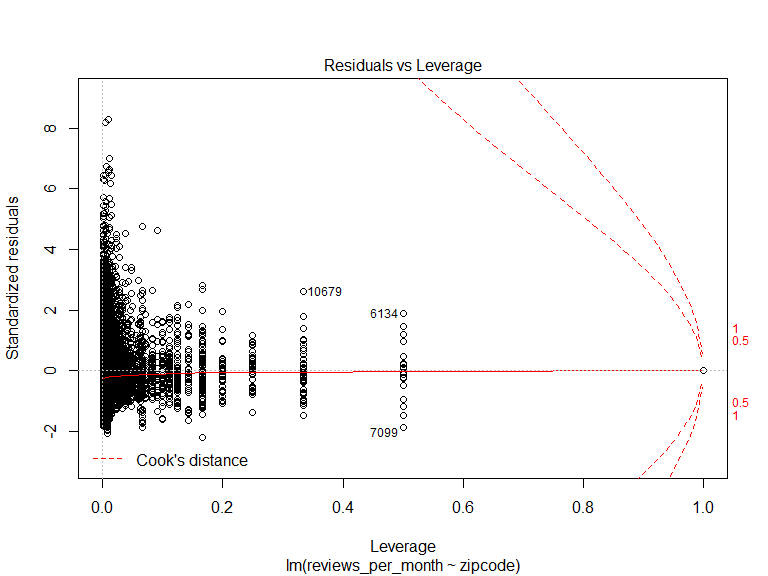


## Warning: not plotting observations with leverage one:  
## 1826, 2608, 6910, 7042, 7195, 7661, 7719, 8222, 9759



## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced



The first plot, Residuals vs Fitted, is a scatterplot of residuals (observed y - predicted y). We want this to be approximately zero. This plot helps us to assess the assumptions of linearity (is red line a line around y=0?) and homoscedasticity (is the spread of residuals even along the x axis?). This model has linearity, but the residuals have larger positive values than negative. So the model is not entirely homoscedastic.

The second plot, Normal Q-Q, compares the residuals to “ideal” normal observations. We want observations to lie along the 45 degree line in the plot. This model follows fairly closely, but at the extreme theoretical quantiles deviate from the line.

The third plot, Scale-Location, shows square-rooted standardized residual vs predicted value and can help visualize homoscedasticity. We want a horizontal line and equal spread of points. This model has some slope in the red line, but is it flat enough? The spread of points seems fairly even.

The fourth plot, Residuals vs Leverage, helps to find outliers and overly-influential observations. Here we look for values at the upper-right and lower-left portions of the plot. No outliers exceed the 0.5 Cook’s distance.

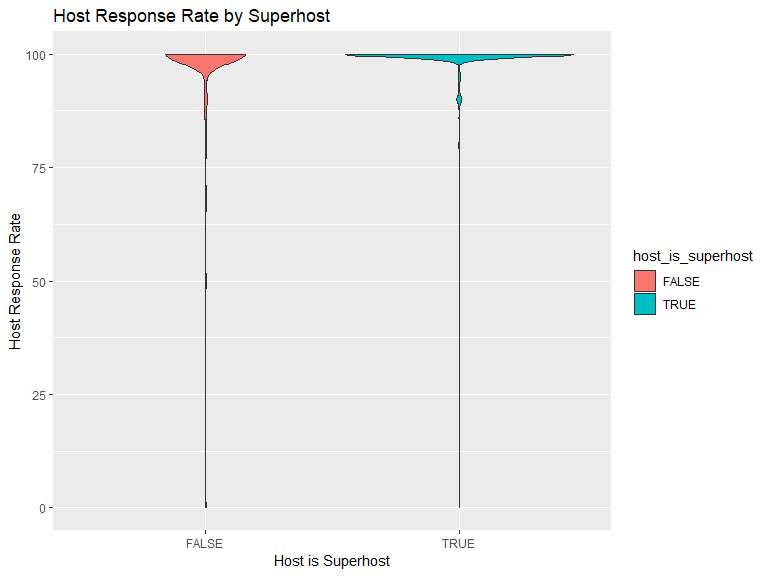
In all, the diagnostic plots of this model show that it meets the assumptions of linearity, normality, and homoscedasticity. I believe this to be a good model, although the correlation level is small.

PART 3: Further Analysis

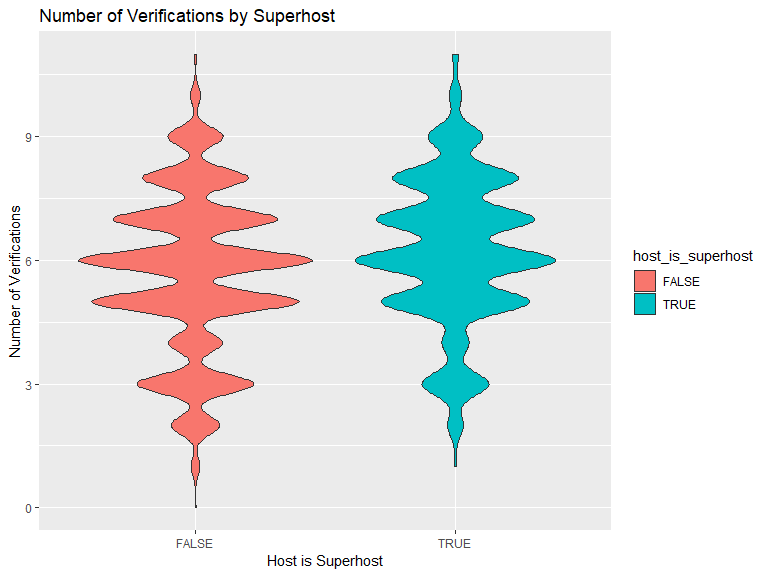
# Q9

# 9.1: Explore relationships (if any) between superhost and host\_since, host\_response\_time, host\_response\_rate. host\_verifications, host\_identity\_verified

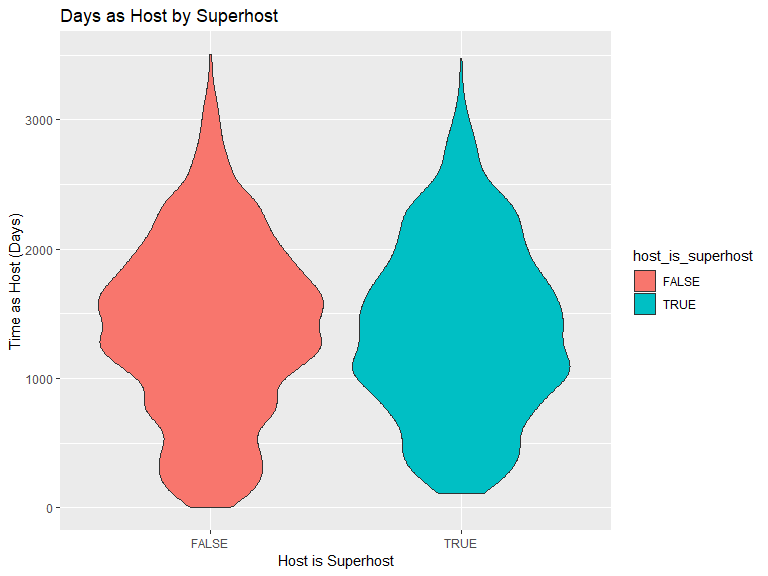
# STEP 1: Make host\_verifications useful  
# Get table of verification counts  
#table(sapply(airbnb$host\_verifications, function(x) length(strsplit(x,",")[[1]])))  
#airbnb %>% select(host\_verifications, number\_of\_verifications) %>% filter(host\_verifications == "[]")  
  
# Split amenities on comma, assign comma count (or 1 if no commas)  
airbnb$number\_of\_verifications <- sapply(airbnb$host\_verifications, function(x) length(strsplit(x,",")[[1]]))  
  
# Clean up the ones; some values of 1 are empty; these are []  
airbnb <- within(airbnb, number\_of\_verifications[host\_verifications == "[]"] <- 0)  
  
  
# Some graphs  
ggplot(airbnb, aes(host\_is\_superhost, host\_response\_rate, fill = host\_is\_superhost)) +  
 geom\_violin() +  
 labs(x = "Host is Superhost", y = "Host Response Rate", title = "Host Response Rate by Superhost")



ggplot(airbnb, aes(host\_is\_superhost, number\_of\_verifications, fill = host\_is\_superhost)) +  
 geom\_violin() +  
 labs(x = "Host is Superhost", y = "Number of Verifications", title = "Number of Verifications by Superhost")



ggplot(airbnb, aes(host\_is\_superhost, host\_number\_of\_days, fill = host\_is\_superhost)) +  
 geom\_violin() +  
 labs(x = "Host is Superhost", y = "Time as Host (Days)", title = "Days as Host by Superhost")



# Some models  
zmodel1 <- lm(host\_response\_rate ~ host\_is\_superhost, data = airbnb)  
summary(zmodel1)

##   
## Call:  
## lm(formula = host\_response\_rate ~ host\_is\_superhost, data = airbnb)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -99.272 0.728 3.480 3.480 3.480   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 96.5200 0.1244 776.16 <2e-16 \*\*\*  
## host\_is\_superhostTRUE 2.7520 0.2446 11.25 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 11.14 on 10813 degrees of freedom  
## Multiple R-squared: 0.01157, Adjusted R-squared: 0.01148   
## F-statistic: 126.6 on 1 and 10813 DF, p-value: < 2.2e-16

#zmodel2 <- lm(host\_response\_time ~ host\_is\_superhost, data = airbnb) # this one is failing  
#summary(zmodel2)  
zmodel3 <- lm(number\_of\_verifications ~ host\_is\_superhost, data = airbnb)  
summary(zmodel3)

##   
## Call:  
## lm(formula = number\_of\_verifications ~ host\_is\_superhost, data = airbnb)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.6845 -1.2111 0.3155 1.3155 5.3155   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.68454 0.02107 269.83 <2e-16 \*\*\*  
## host\_is\_superhostTRUE 0.52655 0.04144 12.71 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.887 on 10813 degrees of freedom  
## Multiple R-squared: 0.01471, Adjusted R-squared: 0.01462   
## F-statistic: 161.5 on 1 and 10813 DF, p-value: < 2.2e-16

zmodel4 <- lm(host\_identity\_verified ~ host\_is\_superhost, data = airbnb)  
summary(zmodel4)

##   
## Call:  
## lm(formula = host\_identity\_verified ~ host\_is\_superhost, data = airbnb)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.5152 -0.4668 -0.4668 0.5332 0.5332   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.466833 0.005574 83.753 < 2e-16 \*\*\*  
## host\_is\_superhostTRUE 0.048373 0.010964 4.412 1.03e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4992 on 10813 degrees of freedom  
## Multiple R-squared: 0.001797, Adjusted R-squared: 0.001705   
## F-statistic: 19.46 on 1 and 10813 DF, p-value: 1.035e-05

zmodel5 <- lm(as.numeric(host\_number\_of\_days) ~ host\_is\_superhost, data = airbnb)  
summary(zmodel5)

##   
## Call:  
## lm(formula = as.numeric(host\_number\_of\_days) ~ host\_is\_superhost,   
## data = airbnb)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1393.64 -478.83 4.99 453.99 2112.36   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1393.642 7.607 183.213 <2e-16 \*\*\*  
## host\_is\_superhostTRUE -15.632 14.963 -1.045 0.296   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 681.2 on 10813 degrees of freedom  
## Multiple R-squared: 0.0001009, Adjusted R-squared: 8.457e-06   
## F-statistic: 1.091 on 1 and 10813 DF, p-value: 0.2962

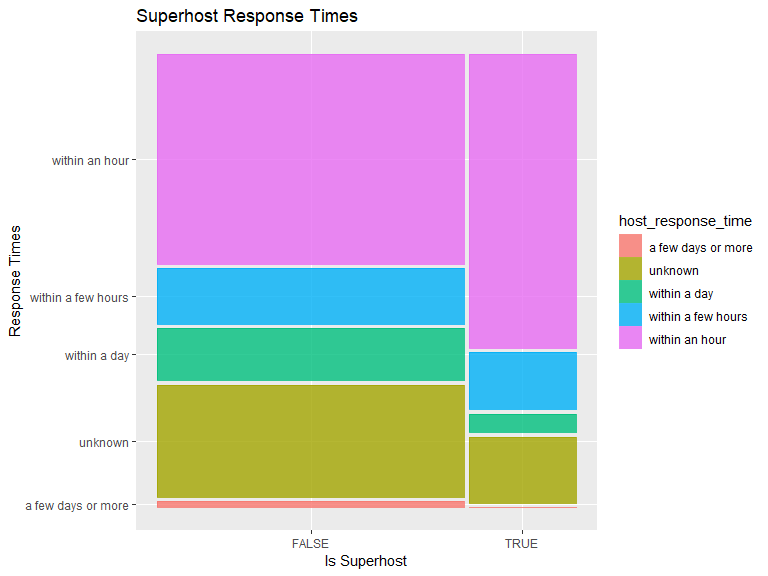
All of these models have small (<0.1) R^2 values. I’m beginning to think I should just burn this project down and start with something else.

# 9.2: Create mosaic plot for host\_response\_time by superhost. What do you learn?

library("ggmosaic")

## Warning: package 'ggmosaic' was built under R version 3.5.2

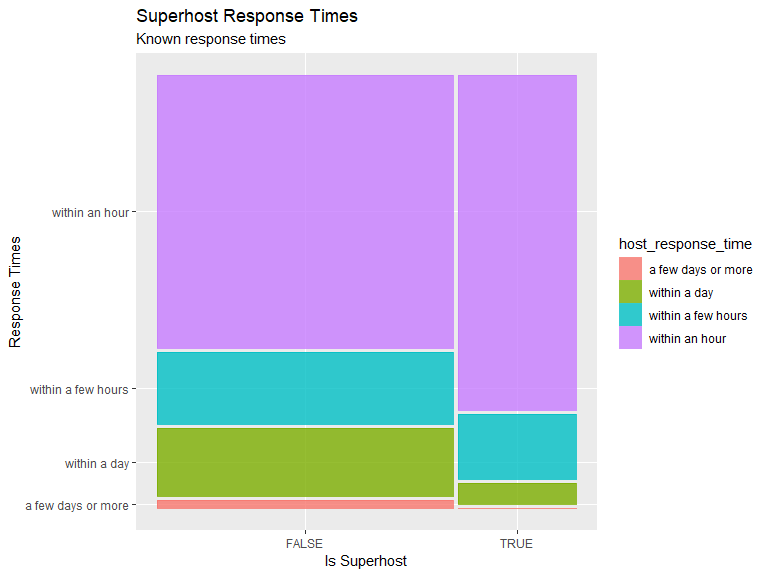
ggplot(data = airbnb) +  
 geom\_mosaic(aes(product(host\_response\_time, host\_is\_superhost), fill=host\_response\_time), na.rm=TRUE) +  
 labs(x="Is Superhost", y="Response Times", title="Superhost Response Times")



# Findings

A mosaic plot shows percentages of observations across categorical variables. Here we can see the distribution of host\_is\_superhost along the x-axis: Superhosts (TRUE) make up less than a third of the total observations (because the FALSE column is more than twice as wide as TRUE). However, we can see the response time distribution along the y-axis, and the data show that over half of Superhosts respond within an hour. This graph includes some imputed data from the cleanup phase. The “unknown” response doesn’t necessarily help us in comparing these two groups (although it does describe the condition of our data). If we want, we can remove “unknown” to compare known response times:

ggplot(data = airbnb %>% filter(host\_response\_time != "unknown")) +  
 geom\_mosaic(aes(product(host\_response\_time, host\_is\_superhost), fill=host\_response\_time), na.rm=TRUE) +  
 labs(x="Is Superhost", y="Response Times", title="Superhost Response Times", subtitle = "Known response times")



Here we can see that, based on known response times, Superhosts respond more quickly than non-Superhosts.

# Q10

# 10.1: Extract unique words in description and eliminate stop words. Store in dataframe and sort decreasing. What do you infer from words with top 10 frequency?

# Get all "words", splitting on space  
#all\_words <- unlist(strsplit(airbnb[1:3,2], "[[:space:]]"))  
# Extract only alpha letters  
#regmatches(all\_words, regexpr("[[:alpha:]]+", all\_words))  
#grep("[[:alpha:]]",unlist(strsplit(airbnb[1:3,2], "[[:space:]]|,|.")), value = TRUE)  
#unlist(grep("[[:alpha:]]", airbnb[1:3,2], value=TRUE))  
  
# After wrangling with "what is a word" and handling punctuation, I've decided to leverage libraries built to deal with this  
# https://www.tidytextmining.com/tidytext.html  
  
#install.packages("tidytext")  
library(tidytext)

## Warning: package 'tidytext' was built under R version 3.5.3

# Load the stop words (to be omitted from analysis) into a tidytext-compatible data frame  
#is457\_stop\_words <- c("a", "able", "about", "across", "after", "all", "almost", "also", "among", "and", "are", "almost", "at", "almost", "also", "am", "among", "an", "and", "any", "are", "as", "at", "be", "because", "been", "but", "by", "can", "cannot", "could", "dear", "did", "do", "does", "either", "else", "ever", "every", "for", "from", "get", "got", "had", "has", "have", "he", "her", "hers", "him", "his", "how", "however", "i", "if", "in", "into", "is", "it", "its", "just", "least", "let", "like", "likely", "may", "me", "might", "most", "must", "my", "neither", "no", "nor", "not", "of", "off", "often", "on", "only", "or", "other", "our", "own", "rather", "said", "say", "says", "she", "should", "since", "so", "some", "than", "that", "the", "their", "them", "then", "there", "these", "they", "this", "is", "to", "too", "was", "us", "wants", "was", "we", "were", "what", "when", "where", "which", "while", "who", "whom", "why", "will", "with", "would", "yet", "you", "your")  
is457\_stop\_words\_df <- data.frame(lexicon = "is457", word = c("a", "able", "about", "across", "after", "all", "almost", "also", "among", "and", "are", "almost", "at", "almost", "also", "am", "among", "an", "and", "any", "are", "as", "at", "be", "because", "been", "but", "by", "can", "cannot", "could", "dear", "did", "do", "does", "either", "else", "ever", "every", "for", "from", "get", "got", "had", "has", "have", "he", "her", "hers", "him", "his", "how", "however", "i", "if", "in", "into", "is", "it", "its", "just", "least", "let", "like", "likely", "may", "me", "might", "most", "must", "my", "neither", "no", "nor", "not", "of", "off", "often", "on", "only", "or", "other", "our", "own", "rather", "said", "say", "says", "she", "should", "since", "so", "some", "than", "that", "the", "their", "them", "then", "there", "these", "they", "this", "is", "to", "too", "was", "us", "wants", "was", "we", "were", "what", "when", "where", "which", "while", "who", "whom", "why", "will", "with", "would", "yet", "you", "your"))  
  
# Subset ID and Description  
text\_df <- airbnb[,1:2]  
  
# Unnest\_tokens does some heavy-lifting: it splits out each word, converts to lowercase  
tidy\_txt <- text\_df %>% unnest\_tokens(word, description)  
  
# Remove ("anti-join") the stop words for this class (tidytext comes with its own dictionaries of stop words)  
tidy\_txt <- tidy\_txt %>% anti\_join(is457\_stop\_words\_df)

## Warning: Column `word` joining character vector and factor, coercing into  
## character vector

# Count'em and keep the top 10  
tidy\_txt %>% count(word, sort = TRUE) %>% head(10)

## # A tibble: 10 x 2  
## word n  
## <chr> <int>  
## 1 apartment 14658  
## 2 bedroom 10692  
## 3 walk 10680  
## 4 sydney 9734  
## 5 room 9702  
## 6 kitchen 9340  
## 7 beach 8965  
## 8 bed 8872  
## 9 2 7575  
## 10 house 7236

# 10.2a: Explore whether beach affects price of a listing. What is the difference in average price?  
  
# Get "beach" and "beaches" listing IDs  
has\_beach <- tidy\_txt$id[tidy\_txt$word == "beach"]  
has\_beaches <- tidy\_txt$id[tidy\_txt$word == "beaches"]  
  
# These vectors have every mention of the word  
paste(length(has\_beach),"mentions of beach, and",length(has\_beaches),"mentions of beaches.")

## [1] "8965 mentions of beach, and 2054 mentions of beaches."

# Show unique listings with these words  
paste(length(unique(has\_beach)),"listings have beach, and",length(unique(has\_beaches)),"listings have beaches.")

## [1] "4016 listings have beach, and 1550 listings have beaches."

# union() removes duplicates  
beachy <- union(has\_beach, has\_beaches)  
paste(length(beachy),"listings in total mention beach or beaches.")

## [1] "4655 listings in total mention beach or beaches."

# How do beach listings prices compare to non-beach?  
# Convert beachy vector to a data frame to merge into airbnb as a logical column  
beachy <- data.frame(id = beachy, beach\_desc = TRUE)  
  
airbnb <- merge(airbnb, beachy, all.x = TRUE, by="id")  
airbnb$beach\_desc[is.na(airbnb$beach\_desc)] <- FALSE  
  
# I still have 4655 listings with beach, now as a logical column in airbnb  
table(airbnb$beach\_desc)

##   
## FALSE TRUE   
## 6160 4655

mean(airbnb$price)

## [1] 203.1568

by(airbnb$price, airbnb$beach\_desc, mean)

## airbnb$beach\_desc: FALSE  
## [1] 177.6529  
## --------------------------------------------------------   
## airbnb$beach\_desc: TRUE  
## [1] 236.9063

airbnb %>% group\_by(beach\_desc) %>% summarise(mean(price))

## # A tibble: 2 x 2  
## beach\_desc `mean(price)`  
## <lgl> <dbl>  
## 1 FALSE 178.  
## 2 TRUE 237.

# Findings

The average price for all listings is $203.16. Listings that mention “beach” or “beaches” average price is $236.91 ($33.75 above average). Listings without those words average $177.25 ($25.51 below average).

Compared to each other, a beach listing is priced 33% higher.

# 10.2b: Explore multiple high frequency words. Write a function to get word frequency by row.

count\_word\_in\_desc <- function(w, v){  
 # funcname: count\_word\_in\_desc  
 # inputs : A word (character string) to search for, a vector to search in  
 # outputs : count of words (integer)  
 # purpose : Count appearances of a string  
 # related : N/A  
 # auth/dt : ID35, 2019-04-25  
   
 w <- tolower(w)  
 v <- tolower(v)  
   
 if(attr(gregexpr(w,v)[[1]], "match.length")[1] == -1){  
 return(0)  
 }  
 else{  
 return(length(attr(gregexpr(w,v)[[1]], "match.length")))  
 }  
}  
  
# Now to find some top words! I dug a little  
# Beach, just for comparison to the tidytext. This one finds 4,694 beach listings, but the results are similar.  
airbnb$wc\_beach <- unlist(lapply(airbnb$description, function(x) count\_word\_in\_desc("beach",x)))  
airbnb %>% group\_by(wc\_beach>0) %>% summarise(n = n(), avg\_price = mean(price), avg\_review = mean(review\_scores\_rating))

## # A tibble: 2 x 4  
## `wc\_beach > 0` n avg\_price avg\_review  
## <lgl> <int> <dbl> <dbl>  
## 1 FALSE 6121 177. 93.8  
## 2 TRUE 4694 237. 94.6

# What listings are different?  
diffs <- airbnb$description[airbnb$beach\_desc == FALSE & airbnb$wc\_beach > 0]  
table(regmatches(diffs, regexpr("beach[[:alpha:]]\*", diffs)))

##   
## beachâ beaché beaches beachfront   
## 1 1 1 4   
## beachside beachvolleyball beachy   
## 19 1 7

# 10.3: Select at least 3 other words from your dataframe and do similar analysis. What conclusions do you find?

airbnb$wc\_kitchen <- unlist(lapply(airbnb$description, function(x) count\_word\_in\_desc("kitchen",x)))  
airbnb$wc\_restaurants <- unlist(lapply(airbnb$description, function(x) count\_word\_in\_desc("restaurants",x)))  
airbnb$wc\_bus <- unlist(lapply(airbnb$description, function(x) count\_word\_in\_desc("bus",x)))  
airbnb$wc\_walk <- unlist(lapply(airbnb$description, function(x) count\_word\_in\_desc("walk",x)))  
airbnb$wc\_balcony <- unlist(lapply(airbnb$description, function(x) count\_word\_in\_desc("balcony",x)))  
  
#airbnb$wc\_kitchen[airbnb$wc\_kitchen > 0]  
airbnb %>% group\_by(wc\_kitchen>0) %>% summarise(n = n(), avg\_price = mean(price), avg\_review = mean(review\_scores\_rating))

## # A tibble: 2 x 4  
## `wc\_kitchen > 0` n avg\_price avg\_review  
## <lgl> <int> <dbl> <dbl>  
## 1 FALSE 3848 211. 94.3  
## 2 TRUE 6967 199. 94.1

airbnb %>% group\_by(wc\_restaurants>0) %>% summarise(n = n(), avg\_price = mean(price), avg\_review = mean(review\_scores\_rating))

## # A tibble: 2 x 4  
## `wc\_restaurants > 0` n avg\_price avg\_review  
## <lgl> <int> <dbl> <dbl>  
## 1 FALSE 6309 214. 94.0  
## 2 TRUE 4506 188. 94.4

airbnb %>% group\_by(wc\_bus>0) %>% summarise(n = n(), avg\_price = mean(price), avg\_review = mean(review\_scores\_rating))

## # A tibble: 2 x 4  
## `wc\_bus > 0` n avg\_price avg\_review  
## <lgl> <int> <dbl> <dbl>  
## 1 FALSE 5422 230. 94.2  
## 2 TRUE 5393 176. 94.2

airbnb %>% group\_by(wc\_walk>0) %>% summarise(n = n(), avg\_price = mean(price), avg\_review = mean(review\_scores\_rating))

## # A tibble: 2 x 4  
## `wc\_walk > 0` n avg\_price avg\_review  
## <lgl> <int> <dbl> <dbl>  
## 1 FALSE 3610 222. 93.9  
## 2 TRUE 7205 194. 94.3

airbnb %>% group\_by(wc\_balcony>0) %>% summarise(n = n(), avg\_price = mean(price), avg\_review = mean(review\_scores\_rating))

## # A tibble: 2 x 4  
## `wc\_balcony > 0` n avg\_price avg\_review  
## <lgl> <int> <dbl> <dbl>  
## 1 FALSE 7812 207. 94.1  
## 2 TRUE 3003 193. 94.4

## Findings

Scanning the top words, I looked for words that indicated something possibly unique about a listing (e.g. kitchen) versus something more routine (e.g., bed). I focused on a few areas: food (kitchen vs restaurants), transportation (bus vs walking), and the balcony (just curious about it more than anything).

### Effect on Review Score

What I found is that these words had a small effect on the average review (the largest spread was 0.4, or just under a quarter-star difference: “walk” and “restaurants” both scored higher than their non-walk and non-restaurant counterparts).

### Effect on Price

Food: Presence of either “kitchen” or “restaurant” is tied to a lower average price. Kitchen is $12 less than non-kitchen, and restaurant is $26 less than non-restaurant.

Transportation: Presence of “bus” and “walk” are both linked to lower average price. Bus is $54 less than non-bus, and walk is $28 less than non-walk. It seems that nobody really wants to think about taking the bus.

Balcony: Presence of “balcony” is also related to lower average price – $14 less if the description mentions “balcony”.

# Q10(2)

# Q10(2).1 Choose between zip code or city. Justify. Calculate number of listings for each in category. Filter to top 100.

**Explore whether top 100 have higher weighted ratings. Graph and explain your findings.**

The city data is very messy, with case and formatting issues. I’m going to use zip code data.

# Grab the top 100, but exclude the "unknown" that I imputed earlier.  
#head(sort(table(airbnb$zipcode[airbnb$zipcode != "unknown"]), decreasing = TRUE), 100)  
  
# I have just been learning the dplyr tools with this project, and I find this much more readable:  
# How many listings in the top 100?  
airbnb %>% select(zipcode) %>% filter(zipcode != "unknown") %>% table() %>% sort(decreasing = TRUE) %>% head(100) %>% sum()

## [1] 10291

# Filter the top 100 zipcodes  
zip100 <- airbnb %>% select(zipcode) %>% filter(zipcode != "unknown") %>% table() %>% sort(decreasing = TRUE) %>% head(100) %>% row.names()  
  
airbnb %>% filter(zipcode %in% zip100) %>% select(c(zipcode, review\_scores\_rating, number\_of\_reviews)) %>% group\_by(zipcode) %>% summarise(wmean = weighted.mean(review\_scores\_rating, number\_of\_reviews))

## # A tibble: 100 x 2  
## zipcode wmean  
## <chr> <dbl>  
## 1 2000 93.1  
## 2 2007 93.3  
## 3 2008 92.9  
## 4 2009 93.9  
## 5 2010 93.9  
## 6 2011 94.4  
## 7 2015 96.3  
## 8 2016 94.1  
## 9 2017 94.2  
## 10 2018 95.0  
## # ... with 90 more rows

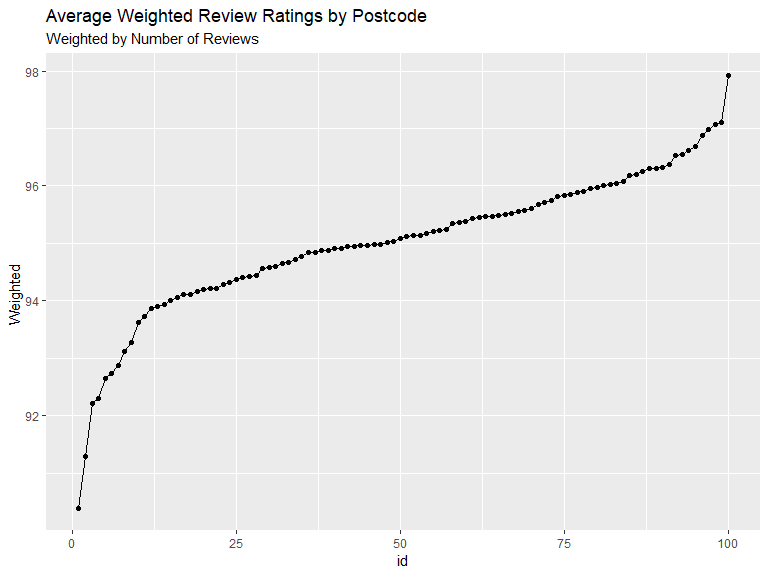
# How many observations (listings) are not in the top 100 postal codes?  
airbnb %>% filter(!(zipcode %in% zip100)) %>% count()

## # A tibble: 1 x 1  
## n  
## <int>  
## 1 524

This makes little sense. The top 100 zipcodes is most of the data set – 10,291 of the 10,815 observations. Can I reduce this number, like show the top 50? Am I supposed to use the inconsistently-named cities? Or am I actually to compare 10,291 to 524 observations?

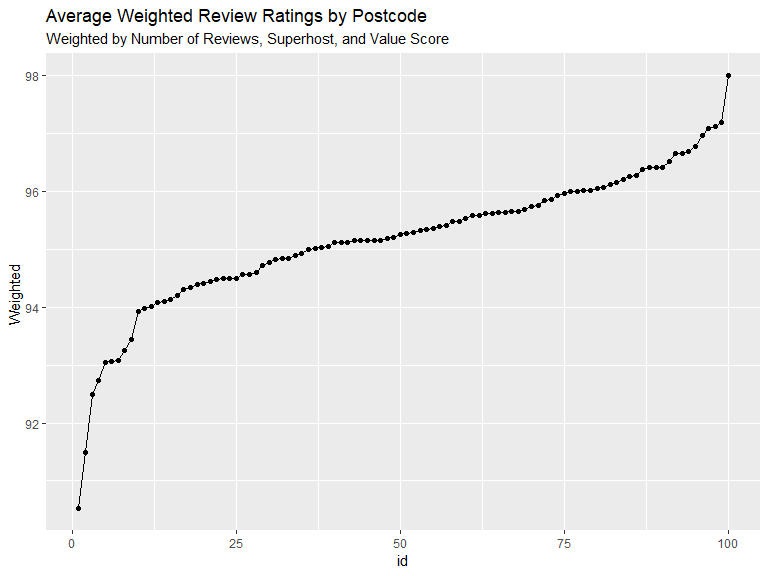
Instead, I will show the trend of the top 100, putting each of the 100 zipcodes on the x-axis and showing the weighted mean.

wmeans <- airbnb %>% filter(zipcode %in% zip100) %>% select(c(zipcode, review\_scores\_rating, number\_of\_reviews)) %>% group\_by(zipcode) %>% summarise(wmean = weighted.mean(review\_scores\_rating, number\_of\_reviews))  
wmeans <- sort(wmeans$wmean)  
  
wmeans <- data.frame(wmean = wmeans) # make a data.frame  
wmeans$id <- as.numeric(row.names(wmeans)) # add an id  
  
ggplot(wmeans, aes(x = id, y = wmean)) +  
 geom\_point() +  
 geom\_line() +  
 labs("x = ID", y = "Weighted ", title = "Average Weighted Review Ratings by Postcode", subtitle = "Weighted by Number of Reviews")



# Q10(2).2: Choose two other aspects from description that may improve the weighted mean of review\_scores\_rating

# Superhosts should count for twice as much as regular hosts  
# as.numeric(airbnb$host\_is\_superhost) + 1 # I dunno, this was breaking the weighted mean  
# Let's emphasize value  
  
  
wmeans <- airbnb %>% filter(zipcode %in% zip100) %>% select(c(zipcode, review\_scores\_rating, number\_of\_reviews, host\_is\_superhost, review\_scores\_value)) %>% group\_by(zipcode) %>% summarise(wmean = weighted.mean(review\_scores\_rating, (number\_of\_reviews \* review\_scores\_value)))  
wmeans <- sort(wmeans$wmean)  
  
wmeans <- data.frame(wmean = wmeans) # make a data.frame  
wmeans$id <- as.numeric(row.names(wmeans)) # add an id  
  
ggplot(wmeans, aes(x = id, y = wmean)) +  
 geom\_point() +  
 geom\_line() +  
 labs("x = ID", y = "Weighted ", title = "Average Weighted Review Ratings by Postcode", subtitle = "Weighted by Number of Reviews, Superhost, and Value Score")



I don’t think I chose the right variables to add to the graph, and the Superhost weighted value was breaking for me. What stands out is that most of the average scores are between 94 and 97. Extreme values are likely caused by lower n values.

PART 4: Your Turn

# Conduct further analysis

#install.packages(c("sp", "rgdal"))  
library(sp)

## Warning: package 'sp' was built under R version 3.5.3

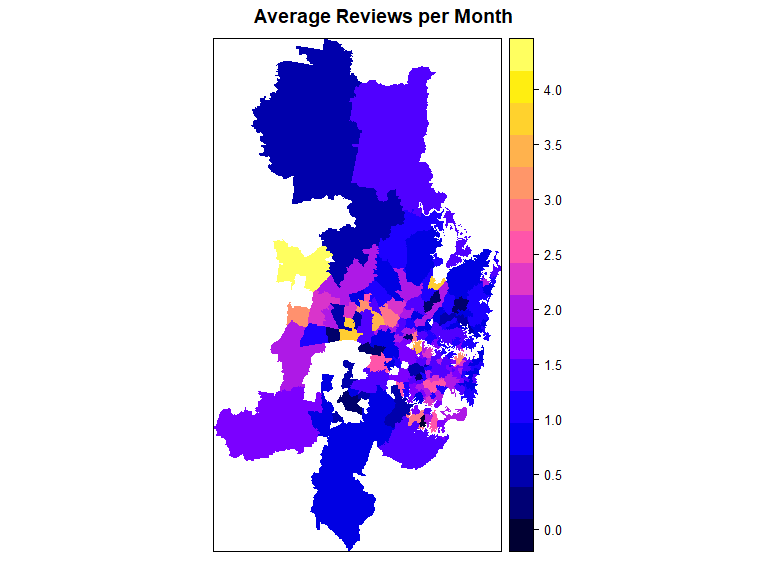
library(rgdal)

## Warning: package 'rgdal' was built under R version 3.5.3

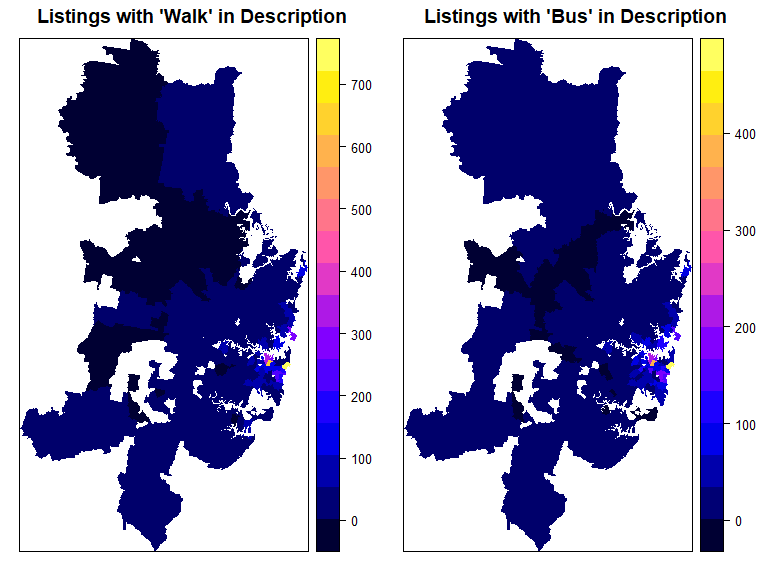
# Postal Areas ASGS Ed 2016 Digital Boundaries in ESRI Shapefile Format  
# Shapefile from http://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/1270.0.55.003July%202016?OpenDocument  
# direct link: http://www.abs.gov.au/ausstats/subscriber.nsf/log?openagent&1270055003\_poa\_2016\_aust\_shape.zip&1270.0.55.003&Data%20Cubes&4FB811FA48EECA7ACA25802C001432D0&0&July%202016&13.09.2016&Previous  
# Processed in ArcGIS to inner join postal codes  
# Post-processed data available for download at: https://github.com/mapsquatch/IS457/tree/master/data  
shape <- readOGR(dsn = ".\\data", layer = "sydneyGIS")

## OGR data source with driver: ESRI Shapefile   
## Source: "C:\Users\Phil\Documents\IS457\data", layer: "sydneyGIS"  
## with 203 features  
## It has 5 fields

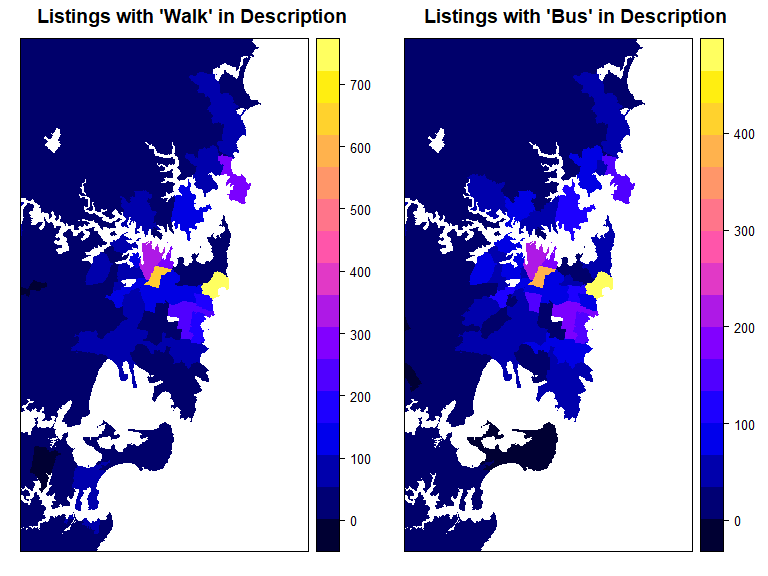
# de-factor the merge field  
shape$POA\_CODE16 <- as.character(shape$POA\_CODE16)  
  
# Summarize by postcode; mutate new variables for NUMBER of listings mentioning walk or bus (the word counts could do multiples per listing)  
attrtable <- airbnb %>% mutate(n\_walk = wc\_walk > 0, n\_bus = wc\_bus > 0) %>% group\_by(zipcode) %>% summarise(n = n(), walk = sum(n\_walk), bus = sum(n\_bus), avg\_rev\_mo = mean(reviews\_per\_month))  
  
sydneylyr <- merge(shape, attrtable, by.x = "POA\_CODE16", by.y = "zipcode")  
  
# Plot Average Reviews Per Month  
spplot(sydneylyr, "avg\_rev\_mo", main = "Average Reviews per Month", col = "transparent")



# Side by side plot  
# Plot Walk Mentions  
sp\_out\_walk <- spplot(sydneylyr, "walk", main = "Listings with 'Walk' in Description", col = "transparent")  
print(sp\_out\_walk, split=c(1, 1, 2, 1), more=TRUE)  
# Plot Bus Mentions  
sp\_out\_bus <- spplot(sydneylyr, "bus", main = "Listings with 'Bus' in Description", col = "transparent")  
print(sp\_out\_bus, split=c(2, 1, 2, 1), more=FALSE)



# Now zoom in to harbor area  
scale.parameter = 0.3 # scaling parameter. less than 1 is zooming in, more than 1 zooming out.   
xshift = 0.35 # Shift to right in map units.   
yshift = -0.2 # Shift to left in map units.   
original.bbox = sydneylyr@bbox # Pass bbox of your Spatial\* Object.   
  
edges = original.bbox  
edges[1, ] <- (edges[1, ] - mean(edges[1, ])) \* scale.parameter + mean(edges[1, ]) + xshift  
edges[2, ] <- (edges[2, ] - mean(edges[2, ])) \* scale.parameter + mean(edges[2, ]) + yshift  
   
# Plot Walk Mentions - ZOOM IN  
sp\_out\_walk <- spplot(sydneylyr, "walk", main = "Listings with 'Walk' in Description", col = "transparent", xlim = edges[1, ], ylim = edges[2, ])  
print(sp\_out\_walk, split=c(1, 1, 2, 1), more=TRUE)  
# Plot Bus Mentions - ZOOM IN  
sp\_out\_bus <- spplot(sydneylyr, "bus", main = "Listings with 'Bus' in Description", col = "transparent", xlim = edges[1, ], ylim = edges[2, ])  
print(sp\_out\_bus, split=c(2, 1, 2, 1), more=FALSE)



# Findings

To start with, postal codes are dynamic features that change over time. These are from 2016, which is within the date range of our analysis, and thus a suitable dataset for use.

I wanted to further explore the transportation options in the description. My hypothesis was that there would be more mentions of bus in postcodes a little further from the most central.

My methodology was to take the word counts of each target word in the description, and calculate (mutate) a logical variable for analysis that would contain TRUE if the listing contained the target word. Then, summing the raw counts, display them on the map.

I also attempted this with normalizing the data by listings, so it created a density – or percent of listings in a postcode containing the target words. However, this was skewed by the more rural postcodes with fewer listings, and the map appeared a mess.

In using the raw counts, there is an inherent bias where postcodes with more listings will have more listings containing the target words, assuming an even distribution. In effect, this ultimately shows the postcodes with the most listings.

However, the side by side choropleth maps still provide some value, because the color ramp is symbolized relative to each target word, and we can compare the colors to detect any patterns.

In this analysis, I see higher values for “bus” through the periphery (the medium shades of blue), and thus my hypothesis seems to be supported, but not strongly. More statistical analysis is required (but impossible due to time contstraints).

PART 5: Conclusion

Most of the techniques and analyses employed in this project failed to find strong correlations. However, there were a few observations that stood out:

Airbnb hosts looking to upgrade their properties may be interested to know that a cable TV subscription or BBQ grill are both correlated with a higher price than a swimming pool.

Guests may not want to think of logistics when browsing Airbnb. Descriptions containing the following words all had lower prices than those that did not: walk, bus, restaurants, kitchen.

On the contrary, expect to pay (or charge) more beach properties. Listings with “beach” or “beaches” in the listing had prices 33% higher on average.

As a guest, you can expect quicker responses from an Airbnb Superhost.

Check those cleaning fees! You may be surprised to know that nearly 1 in 12 properties charge a cleaning fee that is greater than the price per night.

Final note: For this project, I wanted to learn to use Github with RStudio. You can view any of my files or check my commit history at <https://github.com/mapsquatch/IS457>. I found it to be rather easy to use.

PART 6: Lifecycle of Data Science

The Lifecycle of Data Science refers to the entire process of gathering information from data. Donoho breaks this process down into the six divisions of Greater Data Science (Donoho, 2017). We were able to experience these divisions in this project and in this class.

**GDS1: Data Exploration and Preparation.**

This step is also called “exploratory data analysis,” and it often the most time-intensive task – often estimated as taking 80% of the time. During this step it is the analyst’s job to understand the dataset – both the data and the metadata. For this project, we first had to understand the when (2010-2018), where (Sydney, Australia), why (lodging), who(users: hosts and guests), how (user-collected), and what (data about the property, and the users’ self-reported experience).

Additionally we had to analyze the integrity of these data and plan to deal with missing or faulty data. We did have NA values, but the data were consistent for the most part.

I would add that the 80% of time might be relevant for someone who was adept with the software tools used for later steps, but R is still new to me, and I feel I spent a lot of time fighting to figure out a command or parameter to get things to render correctly.

I would have welcomed a more complete metadata document, but the field\_name and descriptions on the project assignment were enough to get started.

**GDS2: Data Representation and Transformation.**

Data scientists must always consider the format of data. This project employed heavy use of the data frame in R. New variables were added to the data frame, ensuring one observation for each listing. I found that factors generally created more problems than they solved, so I made sure to cast them to numerics or characters whenever appropriate.

Using built-in functions to calculate new data was also important. I used time functions to calculate a number based on days elapsed. I also used regular expressions to parse data from fields of comma-separated values (amenities and verifications).

In hindsight, I could have possibly found more relationships by transforming data using log() or other methods.

**GDS3: Computing with Data.**

The data were provided as a comma-separated value dataset, which is stored in plaintext and is a common format. Data were processed in R, with spatial data coming from ArcGIS.

**GDS4: Data Visualization and Presentation.**

This project relied heavily on data visualization. I used many different types of graphs, and I feel that I’m still learning which ones work best. All semester long we’ve worked with Base R, so I viewed it as an opportunity to learn to work with ggplot2. I think by default ggplot graphs look a little cleaner. I also like the syntax for creating the graphs, although it takes a little while to learn. Only after creating some lattice graphics did I discover facets in ggplot. I would like to work with those in the future.

Building a good graph can be a very time-consuming process, but it is nice that it is repeatable once it is built.

**GDS5: Data Modeling.**

This division is dealing with the two major schools of thought: generative modeling based on existing data (more traditional), and predictive modeling (such as machine learning). Analysis in this project was descriptive, and any use of statistics to predict was done on the assumption that the future conditions would the same as the past.

**GDS6: Science about Data Science.**

This division refers to how data scientists are doing data science. In this project we employed various methodologies, but they were all rather basic. As I learned fundamentals, I did not explore different workflows in the context of what others were doing. I was simply trying to get results.

## References

Donoho, D. (2017). 50 Years of Data Science. *Journal of Computational and Graphical Statistics*, *26*(4), 745–766. https://doi.org/10.1080/10618600.2017.1384734