**R project-Project Revenue Optimization at Zillow-EDA**

——做proj的大体流程和best practice

多思考为什么要做这一步而不是code本身

打开R studio，先用setwd(“your folder path”)把working directory设好，比如数据下载在哪个folder里了，这样所有的output和input会在一个folder里面。

用getwd()可以看到现在是在哪个folder里面。

Setwd(“ ”)报错原因：经常在windows中需要将左斜线改成右斜线或者两个左斜线——R中常见问题

<https://cran.r-project.org/doc/manuals/r-patched/R-intro.html>

An introduction to R

<https://swcarpentry.github.io/r-novice-inflammation/04-cond/>

面试题/take home/公司project：一拿到**先去读description，先去understand problem**，project让你做什么——这个project就是要提高zestimate的准确度，降低error

做proj的**第二步，understand 有什么data可以用**，data长什么样——**data description**

看名为“**data dictionary**”的文件，特别重要，看data是怎么define的——平时做proj一定要看**每个data的definition**，不看的话不知道data对不对

很多new grad在做take home challenge时常犯的错误就是没有understand data,没有做exploratary data analysis，或者data cleaning, data processing,直接build fancy models，这样的candidate会被直接拒掉，因为不understand data时建的模也很可能是错误的。

EDA很重要，一般在工作中占到50%的时间。

今天只做EDA,下次课做model。R coding & project思路

* Loading data的最常用function：

read.csv(file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE, comment.char = "", ...)

File名是必须给的，其他的为默认值，需要改的时候查文档改

Csv file默认用“，”分隔，有的文件也用下划线，就设置sep=’\_’

quote是有的data里有一些特殊字符，可以加一些option来处理

StringsAsFactors:如果数据中有strings，是否要把它当成factor？False，一般不太prefer当成factor。

train <- read.csv("train\_2016\_v2.csv", stringsAsFactors = FALSE)

property <- read.csv("properties\_2016.csv", stringsAsFactors = FALSE)

不了解一个function时加个？直接搜，加？？是globally地搜

* Understand the data：

1. read data descriptions
2. Look at sample data ——有几个常用的functions

**summary**(train)——**如何快速的看一个table里面variable的distribution**

str(train)

dim(train)

head(train)——看data的前几行，看几个samples（看了structure和直接看data的直观感受还是很不一样的）

tail(train)——看倒数几行

colnames(train)

看完train再看下property，一般工作中这几步做的很快但是一定不能缺，take home代码中也必须有这些code

* train表和property表都有parcelid这一列，想要**combine这两个表**：

（1）首先看下这两个表各自**有多少个不重复的id**：

> length(**unique**(train$parcelid))

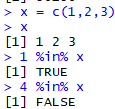
[1] 90150

> length(unique(property$parcelid))

[1] 2985217

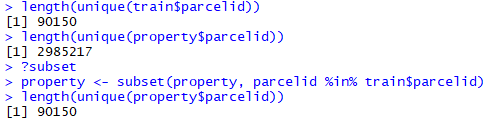
1. 现在在property中做一个**subset**，因为我只关心property这个表中有哪些parcelid在training data中，（**提取**出**要训练的数据的相关properties**），不在training data中的数据就不要了。

property <- subset(property, parcelid %in% train$parcelid)——A %in% B：检查A是否在B中



Subset是R中取子集的常用函数，逗号前面这个parameter是你要对之subset的原数据集的名字，逗号后面的是一个condition。比如subset(train, logerror > 0.5)，就会给出一些例子

现在的property就只有和train一样数目的parcelid了：



1. merge两个table

train <- merge(x = train, y = property, by = 'parcelid', all.x = T) 要merge起来的对象1/df1，对象2/df2，通过哪一列合起来。两个表里的merge列**一样的时候就用by**。这里通过parcelid来merge，但是有时候**两个表里的id名（要merge的那列）不一样**，可以设置merge函数里面的options：**by.x = ??? by.y = ???** 来merge。 “all”option表示要不要保留所有的行，如果有的行在x里，不在y里，是否都要保留？(默认是false)。三个options：

All——是否保留所有行

All.x——是否保留x中的所有行

All.y——是否保留y中的所有行

在这个case中，只要在training data中就都要保留 所以all.x = True。如果没有在training data中，只在property data里有就会delete掉。（因为all.y 默认=false）

有时函数中parameter 的option名可以省略不写，按照位置对应，比如train <- merge(x = train, y = property, by = 'parcelid', all.x = T)就可以写成train <- merge(train, property, by = 'parcelid', all.x = T)，对应第一个位置就是第一个df，x，第二个位置的对应第二个df，y，但有些option名不可省去，比如all.x

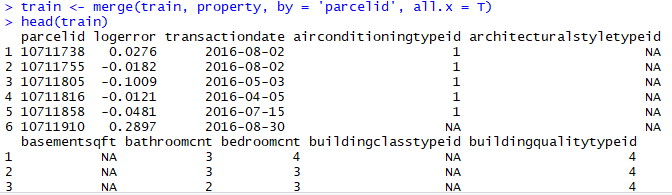
R中的TURE可以简写成T，FALSE可以简写成F。

1. 把merge好的dataframe写到新的csv file 里。（不用每次都操作）

**write.csv**(train, 'train\_property.csv')——写出。（对应read.csv的写入）下次再load就快多了

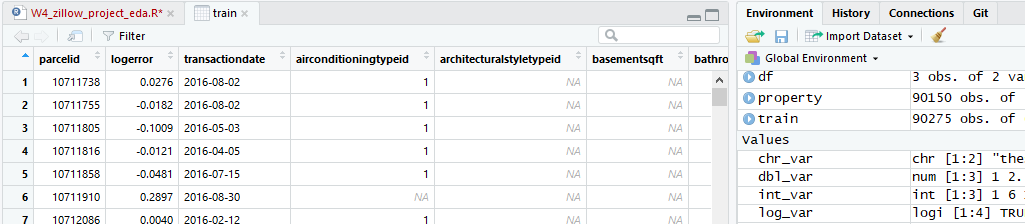
Write.csv写到了当前的work directory里

看一下merge后这个table长什么样：



（前三列是原来的train后面的都是property）

这样看，表有很多列，如果不方便可以直接在右边global environment里点开表格，像excel里面一样比较容易看



Merge两次merge后的文件中会出现明为xxfeature.x, xxfeature.y这样的feature，因为R会把两个merge文件中同名的featue自动命名为feature.x, feature.y，出现了没关系，可以重命名一下，后面讲

为什么要merge——proj让你做什么？用training 做模型。所以重要的是training data中的ID，但是training data只有三列，（transaction date 和 logerror），而其他各种feature都在property这个文件中，要把所有的columns合在一起，所以要去property表中提取我们需要的相关id的信息，combine两个table中共同的information。（不care不在training set中的property中的data。）

* check how many NAs in each feature——处理missing value——非常重要的一步

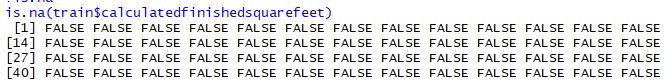
Is.na

length(which(is.na(train$calculatedfinishedsquarefeet)))——看一下**这一列有多少个NA**

**Which** function--index function

Which indices are TRUE? **Give the TRUE indices of a logical object**, allowing for array indices.

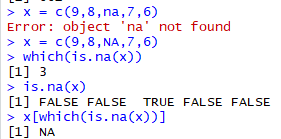
Is.na返回的了一堆T,F，T对应有NA的位置，which把T的位置的indices返回，即Which(is.na)合起来告诉你这一列中有哪些行是missing value，告诉你该列中NA的indices。







经常用which提取一个df中的na：x[which(is.na(x))]



选出其中不是NA的：加个负号——**x[-which(is.na(x))]**



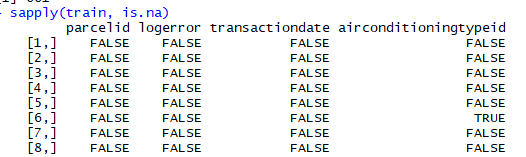
另一种简单一点的数NA的方法——**sum**(**is.na**(train$calculatedfinishedsquarefeet))

R中自动把T算成1，F算成0，如果是NA就是1，不是NA就是0，sum有多少个1即有多少个NA。



\*\*如果我想看**每一列分别有多少个NA**：用**sapply\*\*\*\*\*\*\*\***

sapply(train, FUN = is.na)——**对train这个data set中的每一列，看看na的T/F分布**



**colsums**函数——对这个dataset中的**每个column分别求和**



Colsums和sapply这个组合起来就是看每个column有多少个NA

Sort一下上面的结果：

num.NA <- sort(colSums(sapply(train, is.na)))



# Find out variables with largest number of missing values

**Why do we care about missing values?**

1. Data quality 检查data的质量如何，拿到data第一步先看一下data的质量如何，get a sense. 而missing value特别多的data肯定质量很不好。
2. Missing values reduce the volume of usable data 9万行中6万行是missing的，现在只有3万行能用，大大降低了可用数据量
3. Missing values may lead to biased or even wrong result 做一些东西时要去看missing value能不能用，处理成0还是别的啥数，比如youtube的播放时间这个column，有的用户点开视频只放了个广告，没看视频，播放时间记录为missing，比如这一列里一共100个用户只有两个用户看视频了，一个1h，一个2h，计算视频的平均播放时间，本来应该是（1+2）/100=0.03h.但如果你不管missing value（本该处理为0），直接喂给model，R中model一般自动去掉missing value，计算出来的平均播放时间是（1+2）/2 = 1.5h。这就说明 处理或不处理missing value对结果的影响特别大，两种操作导致的结果差异很大。||||另一种情况，比如想去记录用户的产品生命周期，有两个column：begining day和last day。有的用户1月1日注册后再也没登录，有的用户一直活跃中，活跃用户的last day就是missing。在计算平均寿命时，如果把所有活跃用户的missing都去掉，你相当于只计算了那些不活跃的用户。结果就没有cover到所有用户。这也是一个常见的因为missing data处理不当而biased了result。

R中做一些分析建model时，如果某一行有missing value，只要有NA它就会把这一行自动去掉（不会去掉missing value太多的列/features）。Missing value太多的列不能要，否则R会自动去掉很多行。**我们可以先去掉含太多missing value的列，然后再看一看，采用一些方法填补剩下的少数missing values，以免model把很多数据全部去掉。**

1. 通过看missing values，看你的data是否在采集过程中有问题？比如有些column的data不该missing的却missing，说明数据采集有问题，和data engineer 合作去找出正确data。

上面已经看到每一列中分别有多少missing values，现在我们想找出一些列，它们的missing values不超过该列总行数的20%。

remain.col <- names(num.NA)[which(num.NA <= 0.2 \* dim(train)[1])]

dim(train)[1] 相当于 nrow(train)

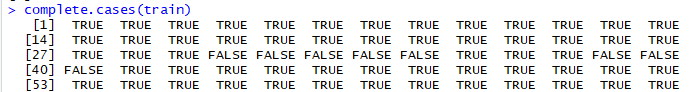
train <- train[, remain.col]

# The percentage of data missing in train.

sum(is.na(train)) / (nrow(train) \*ncol(train))—— 0.007661536

只剩0.76%的missing，还可以

R中还有一个function叫 **complete.cases**(train)，看哪些行是没有任何missing value的



可以看到有82%的行是完全完整的（没有任何missing）

有20%的行有一些missing。

有一些办法来填补missing：面试中非常重要\*\*\*\*\*\*

**\*\*How to treat missing values\*\***

1. -the most important thing before doing all of the following things- Check whether the missing values are random

如果你是Systematically missing-check your data quality!!和data engineer确认一下数据采集过程有无问题。很可能不能用，先查一下数据质量再干活。

Systematically missing——比如所有的male都missing。不能用female的average来代替male的missing。

如果是random missing，才可以做后续的imputation等等。

(1) Remove features with too many missing value, or remove all rows with NA if you **have a lot of data——**只在数据量非常大，且missing是random的才行

(2) add **new level** to represent NA ——把missing value作为一个单独的group，命名为“missing”（给它一个label）作为一个新的level，但是不再是系统里的missing value。单独label出来——category类变量多用

(3) imputation. Example: use library(mice)——数字类变量多用

<https://www.kaggle.com/captcalculator/house-prices-advanced-regression-techniques/imputing-missing-data-with-the-mice-package-in-r/discussion>

(a) take average/median——只要有missing value就让它等于整体的均值

(b) average by group——按组求均值，对不同的组里的missing value，用该组相应的平均值代替

(c) model based imputation——用别的没有missing的features建模来**predict** 这里的missing value是什么

如何判断什么时候用什么missing value的处理方法？前面这两个就是如果有太多missing value就直接去掉这个feature；后面两种方法看情况——对于numerical variable一般imputation做的比较多，结合实践决定，比如连续数字，身高体重这种就做imputation；如果是对categorical variable，如果是学历，房间类型啥的category就可以用new level 代替。

**How do we explore data?**

对不同类型的数据有不同的EDA方法。

1. Numeric variables
2. Categorical variables
3. Numeric variable with response

4) Categorical varialbe with response

1. **Numeric variables——直接看statistics，或者直接看summary（可以对一列求summary）；画一画图（最直观）**

mean(train$logerror)

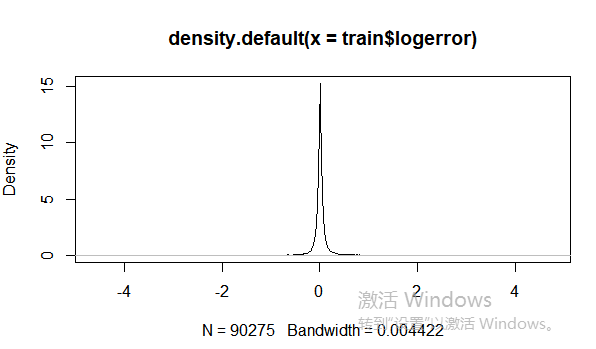
sd(train$logerror)

median(train$logerror)

**quantile(train$logerror, c(0.1, 0.25, 0.5, 0.75, 0.9))**



plot(**density**(train$logerror))——相当于看比例分布



图说明大部分logerror集中在0附近，少部分有一些比较极端的（+-4）

summary(train$logerror)

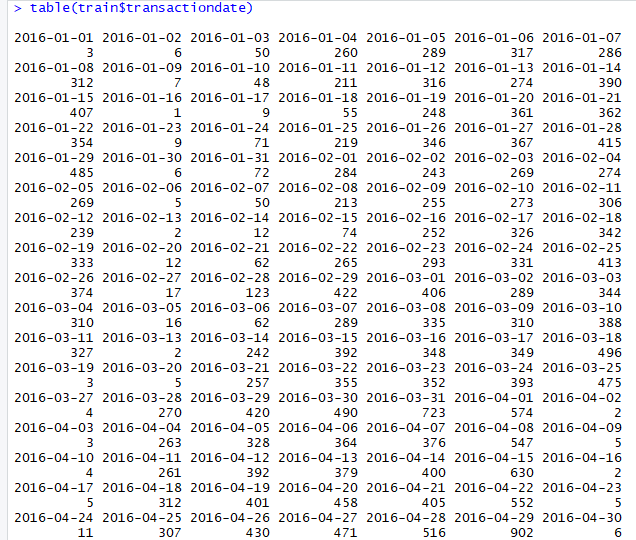


1. **Categorical variables** # Let's check the txndate, character var——内容为字符的变量

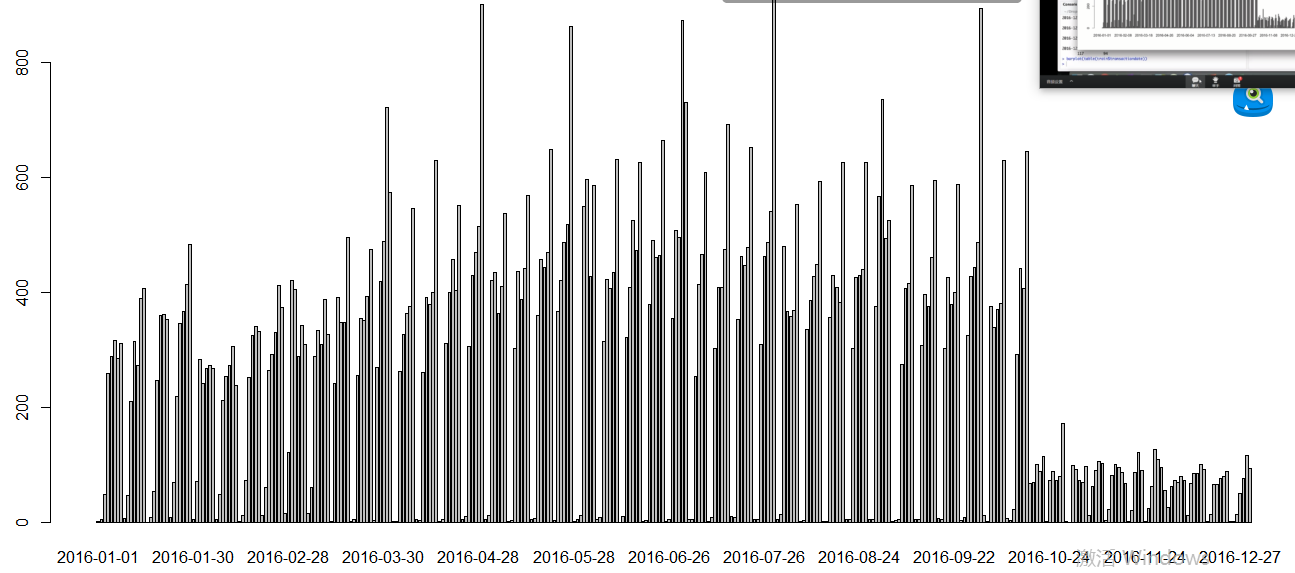
**table**(train$transactiondate)——这个function会自动建个表，把这个column中每个类别对应有多少行统计出来

“table uses the cross-classifying factors to build a contingency table of the **counts at each** combination of factor levels.”

针对这个具体的就是把每个date对应存在多少行数据统计出来，也就是相当于统计**每个date有多少笔交易**。



**barplot**(table(train$transactiondate))——把刚刚那个表统计出来的数据画个barplot

这个图画的就是在每个日期里面产生的transaction数量。

2016年十月之后的数据明显少了是因为description中也说了train-test-split，2016年10月之后的部分数据在test中。

图不是画出来就完事了，画出来图最主要的要**从图中get insights**。

* 从这个图中可以看出**seasonality**——好像发现每个月的**月底** 数据会突然shoot up。月底transaction一般较多。
* 这些空隙代表什么？那些非常低的bar可能是——**weekday的影响**！数一下正好**每7个bar一个波浪**，空隙应该就是周末。

从看出的seasonality，可以对月末加一个flag，加个indicator，月末单独处理；weekday和weekend可能要区分开来，不应该直接用date做variable，可以用day of week来作为一个variable。

**画完图**看到一个现象一定要去想为什么是这样的，**可能的原因**——这也是take home challenge的评分准则。也是没有工作经验的candidates常犯的错误。她把图全画出来，画了十几页的图，却没有一点**insights**，没有自己的想法结论总结。

有31个变量，**都要一个个画图看吗？——NO**

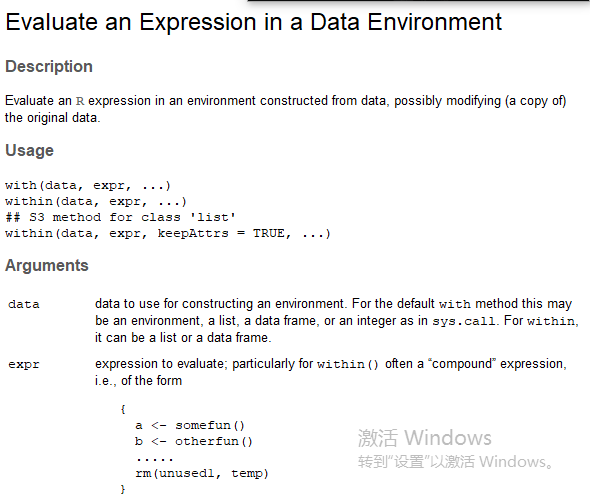
**Business sense**——你觉得哪些variable比较重要的，挑出来画图看。

* 一般**关于时间的variable都非常重要**。因为所有的东西都是跟时间有关的，比如这里我们要predict 2016年10月之后的data，你只有知道前面的时间 数据长什么样你才会有一个sense后面的大概会长什么样。一般**时间的trend一定要看**。
* 要看一下**correlation**. 怎么快速看不同variable之间的correlation？比如我现在最关心的变量是logerror,那我们就看一下那个variable和logerror相关性最高，然后就可以多去花时间专门研究一下这个variable。

画完transactiondate（因为它关于时间），再看一下这个**logerror随着时间的变化有没有不同**？

with(train, plot(as.Date(transactiondate), logerror, **pch = 20**))

**With function**就是取这个“train”data，我后面做的所有的事情都是在“train”这个dataset的基础上做的。



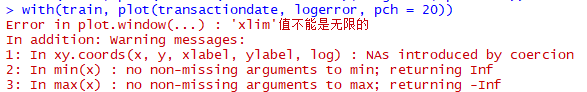
Note that assignments within expr take place in the constructed environment and not in the user's workspace.

这样做的好处就是在后面需要引用某列时，比如as.Date(transactiondate)，就**不需要每次用$指**定是哪个dataset中的这个variable，比如还得写成as.Date(train$transactiondate),**默认都是在train这个dataset中操作**

**as.Date**——Date Conversion Functions to and from Character

Functions to convert between character representations and objects of class "Date" representing calendar dates.

如果不加这个as.Date会报错：



后面的function就是plot transaction date 和logerror。

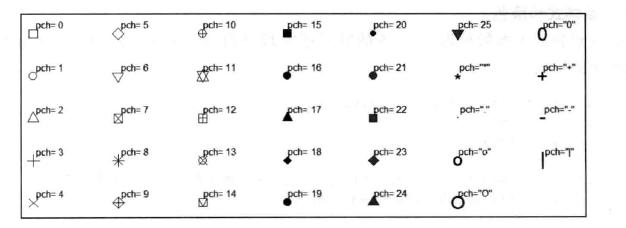
关于”pch”：在R语言中，点的样式由pch的取值决定。

　　当pch取0~14时，其点为空心点，可以用col(颜色)参数设置其边框的颜色；

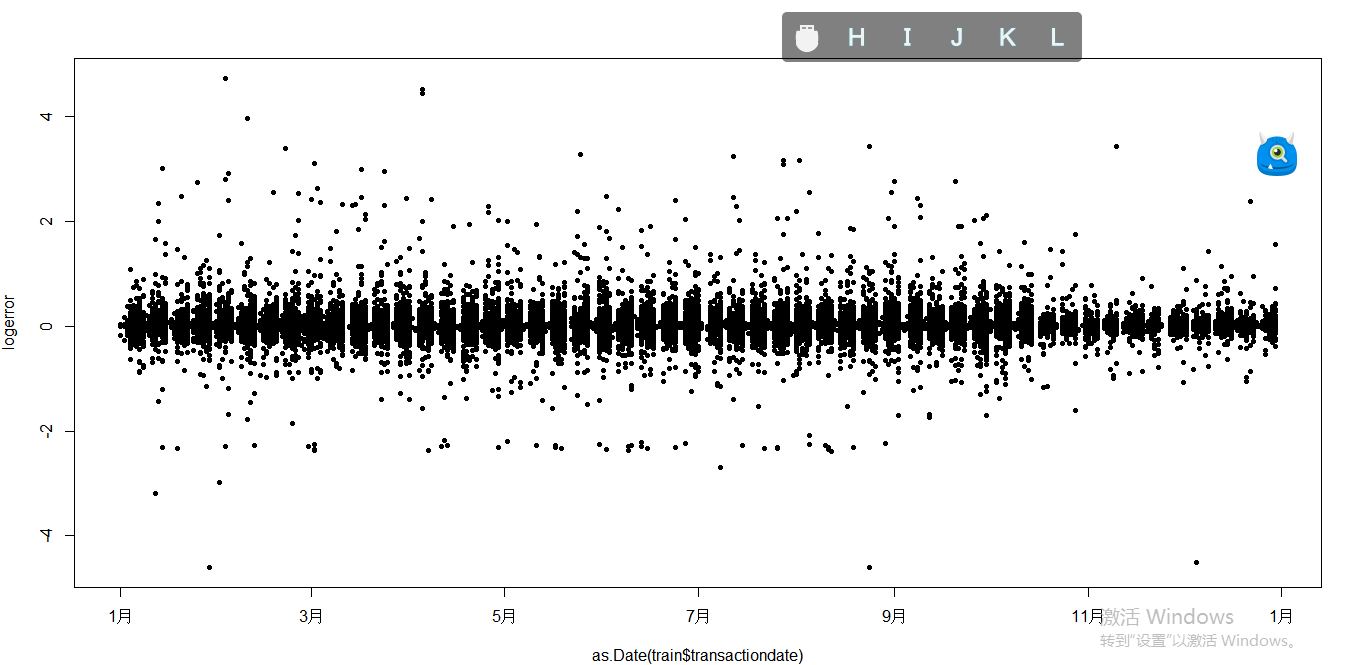
　　当pch取15~20时，其点是实心点，可以用col参数设置其填充的颜色；

　　当pch取21~25时，其点也是实心点，既可以用col参数设置边框的颜色，也可以用bg参数设置其内部的填充颜色。

　　pch不仅包括正方形、圆、上三角、倒三角、菱形等常规图形，还包括之类的特殊图形，如图所示。



发现随着时间的变化lorerror的分布还比较平均，都是在0附近，没有什么trend

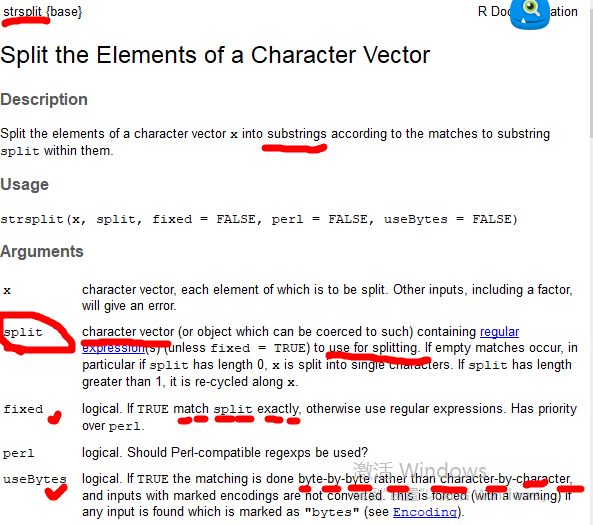


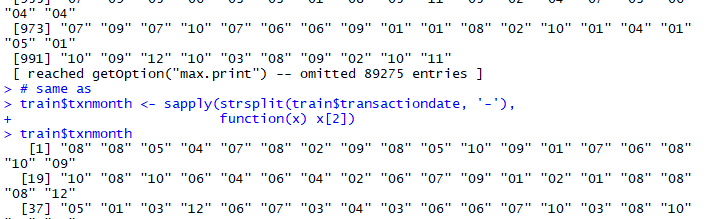
那我们就看一下随着月份的变化有没有什么trend——按月看

train$txnmonth <- sapply(**strsplit(train$transactiondate, '-'**), '[[', **2**)——把transactiondate按“-”分割，会把每个transactiondate拆成年、月、日三个cells，取第二个部分即month赋值给新的变量名叫txnmonth，插入到train这个表格中。Sapply就对每个transactiondate都这样操作（根据-split后取第二部分）。

# same as

train$txnmonth <- sapply(strsplit(train$transactiondate, '-'), function(x) x[2])





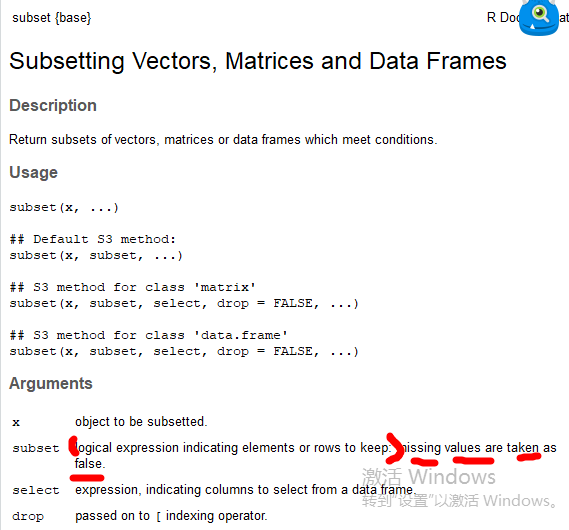
画某几个月的logerror的箱型图：这里画了一月和六月的logerror

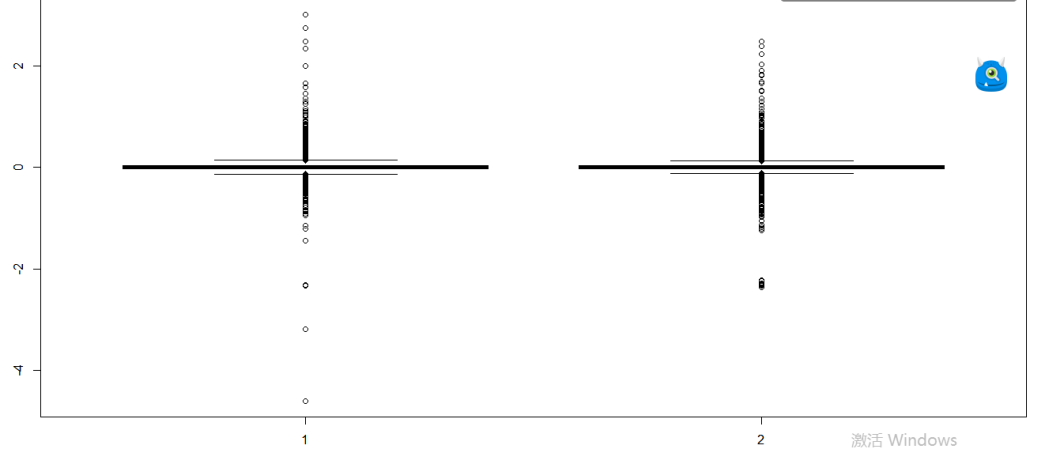
# Q1 - 1.5IQR, Q1, median, Q3, Q3 + 1.5IQR, where IQR is interquartile range: Q3 - Q1

boxplot(subset(train, txnmonth == '01')$logerror,

subset(train, txnmonth == '06')$logerror)

Subset函数

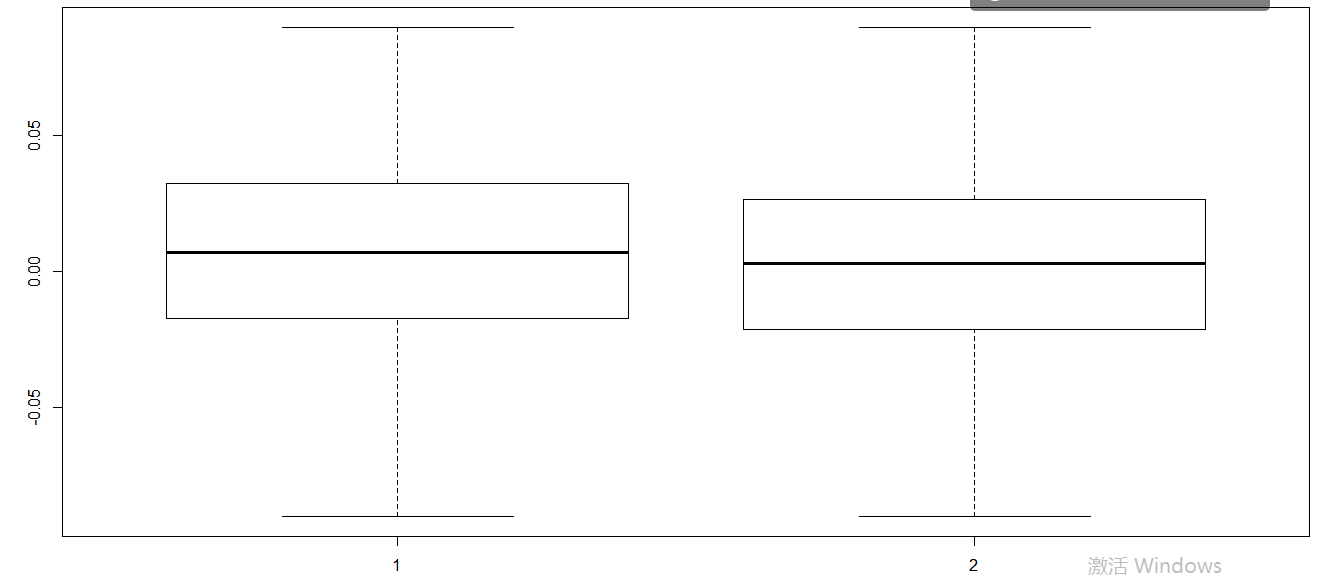
：



这里的1和2指的是level而不是真正的label，可以用这个boxplot中间最黑的线表示median，上下两个线是Q1,Q3，离别的点很远的那些点就是outliers。 这里的outliers太极端以至于中间的点都聚集在一起看不清，我们可以设置只看logerror的绝对值 < 0.09的点：（暂时去掉outliers来看看中间的细节）

boxplot(subset(train, txnmonth == '01' & abs(logerror) < 0.09)$logerror,

subset(train, txnmonth == '06' & abs(logerror) < 0.09)$logerror)



发现1月的logerror的中位数>6月的logerror

——这是基本包里面的，比较两个组（比如1月和6月）的logerror有什么区别。如果我想每个月都比较一下，怎么看？

用library(lattice)

安装新的library的方法：Rstudio->tools->install packages->lattice(搜索你要安装的包)->install

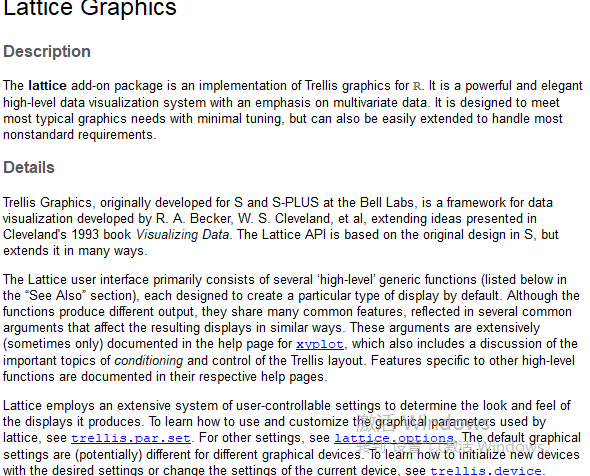
“package ‘lattice’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\hp\AppData\Local\Temp\RtmpEvJWXY\downloaded\_packages”

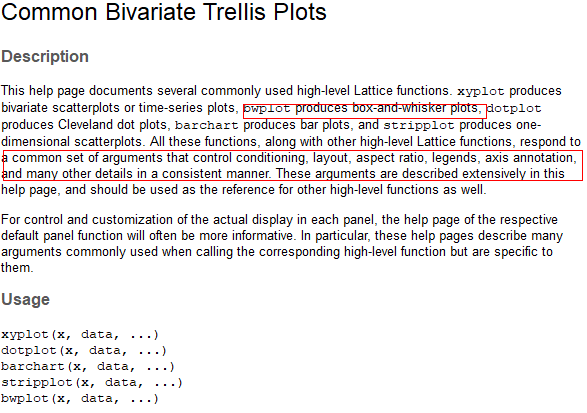
安装完后跑一下library(lattice)——相当于加载这个library

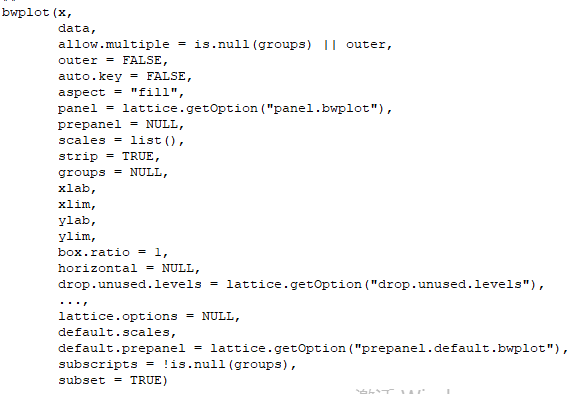
?lattice可以看这个包 其中的bwplot类似于boxplot

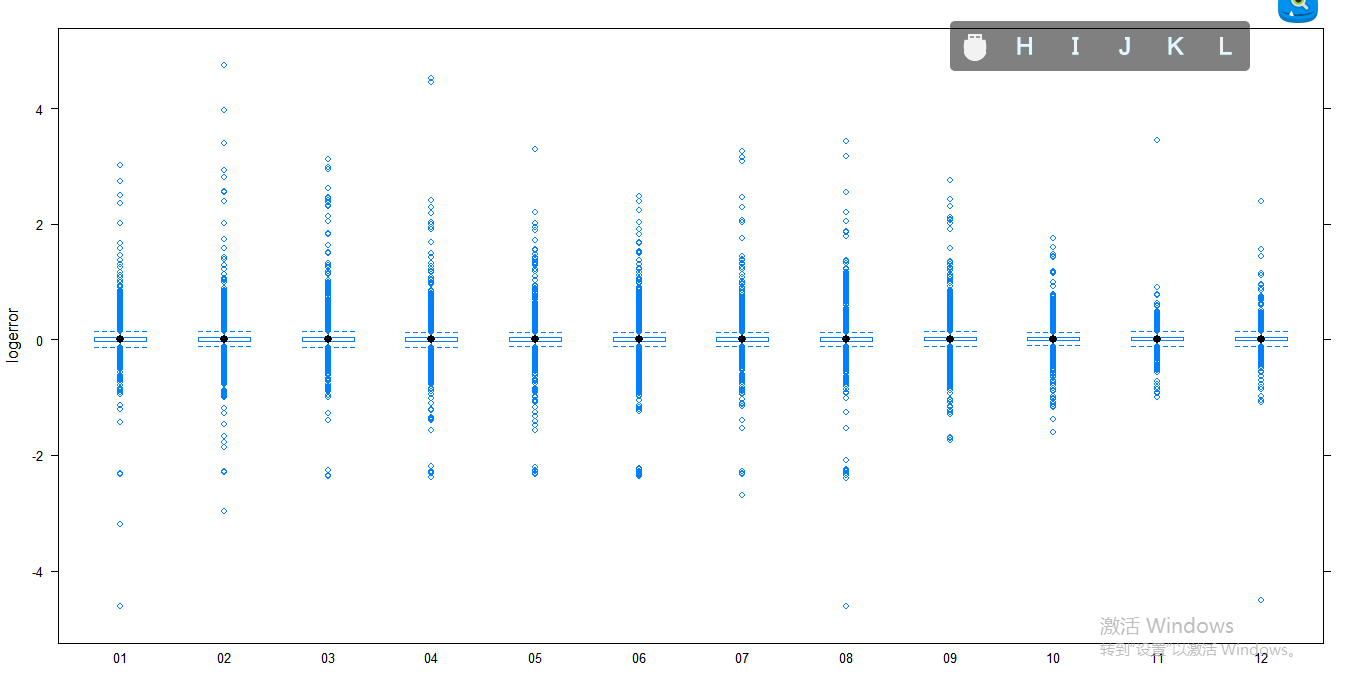


bwplot(logerror ~ txnmonth, data = train)

bwplot(logerror ~ txnmonth, data = subset(train, abs(logerror) < 0.09))

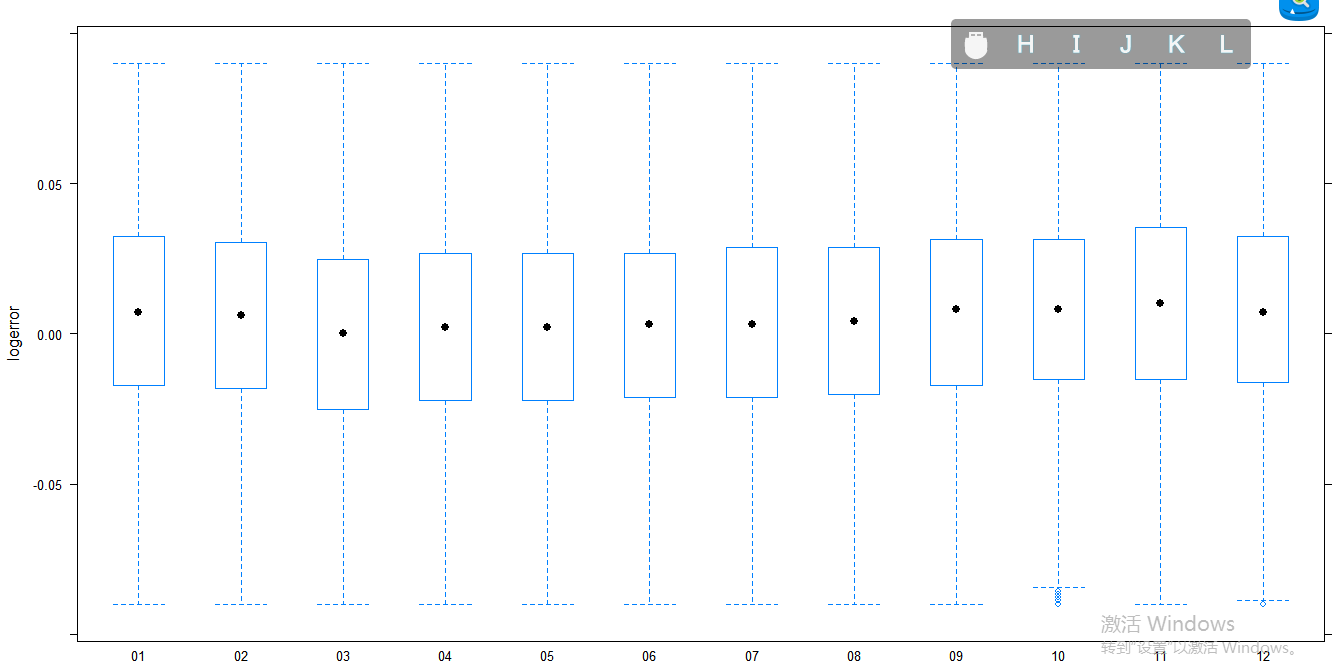






画出了每个月的boxplot

同样的，去掉outliers看中间细节：



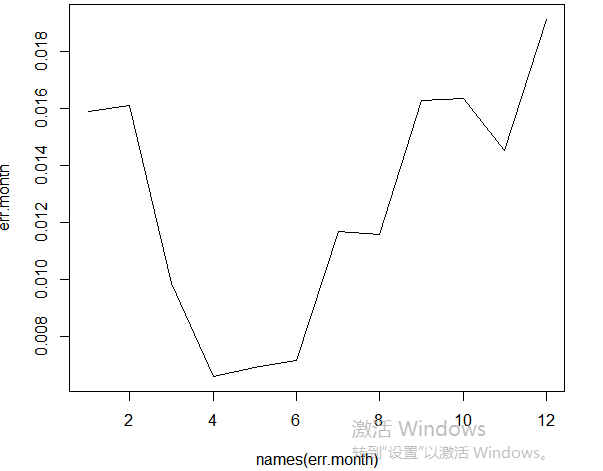
发现每个月有起伏，但差别不太大

接下来看每个月的logerror均值的差异：

err.month <- by(train, train$txnmonth, function(x) {

return(mean(x$logerror))})

plot(names(err.month), err.month, type = 'l')



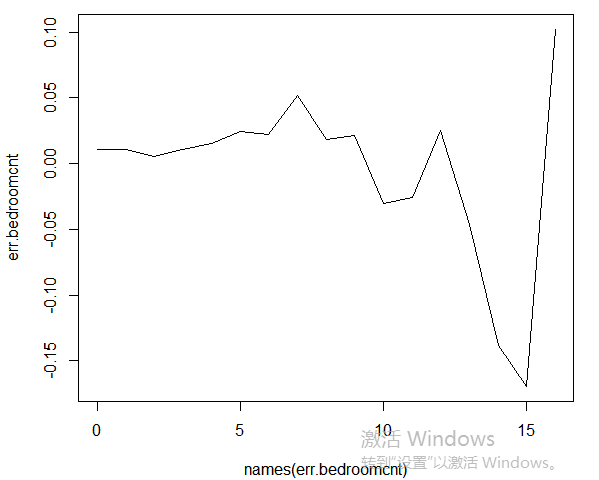
4月平均logerror最小，12月平均logerror最大

每个bedroomcnt类别对应的logerror均值的差异：

err.bedroomcnt <- by(train, train$bedroomcnt, function(x) {

return(mean(x$logerror))})

plot(names(err.bedroomcnt), err.bedroomcnt, type = 'l')



15个bedroom的house价值被underpredicted。

——以上只是举例，该具体看哪个不看哪个？可以通过correlation决定

**Correlation**

看不同features之间的相关性

# check remaining features, some numerical features should be charater type

# Only three counties: (this is fip)

# 6037 Los Angeles

# 6059 Orange County

# 6111 Ventura County

先安装一个“corrplot”的包

package ‘corrplot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\hp\AppData\Local\Temp\RtmpEvJWXY\downloaded\_packages

train[, c('fips', 'propertylandusetypeid', 'rawcensustractandblock', 'regionidcounty', 'assessmentyear', 'regionidzip', 'censustractandblock', 'regionidcity')] <-

as.character(train[, c('fips', 'propertylandusetypeid', 'rawcensustractandblock', 'regionidcounty', 'assessmentyear', 'regionidzip', 'censustractandblock', 'regionidcity')])

Cor是一个计算correlations的function，

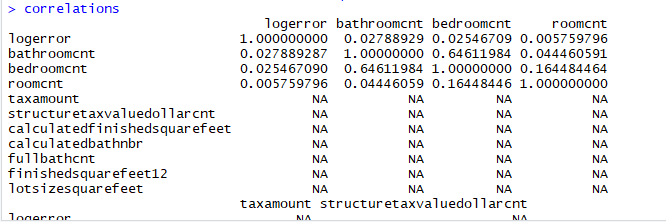
library(corrplot)

correlations <- cor(train[, c('logerror', 'bathroomcnt', 'bedroomcnt', 'roomcnt',

'taxamount', 'structuretaxvaluedollarcnt', 'calculatedfinishedsquarefeet',

'calculatedbathnbr', 'fullbathcnt', 'finishedsquarefeet12',

'lotsizesquarefeet')]) # see a lot of NA

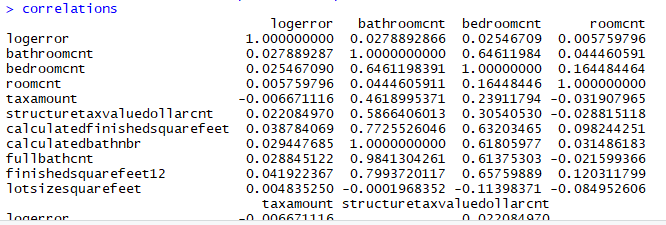


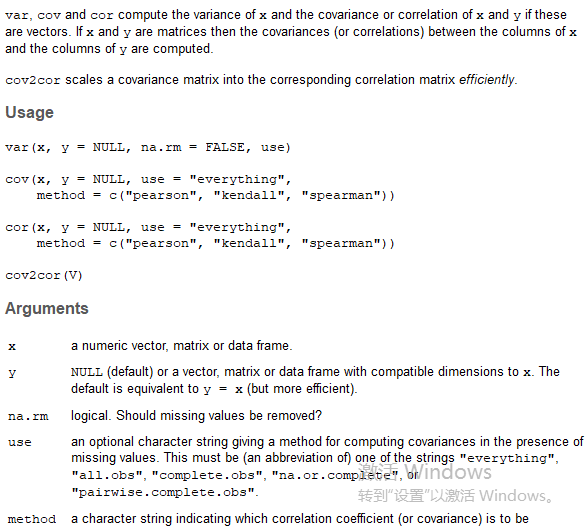
correlations <- cor(train[, c('logerror', 'bathroomcnt', 'bedroomcnt', 'roomcnt',

'taxamount', 'structuretaxvaluedollarcnt', 'calculatedfinishedsquarefeet',

'calculatedbathnbr', 'fullbathcnt', 'finishedsquarefeet12', 'lotsizesquarefeet')],

**use = "pairwise.complete.obs"**)





这些变量都是numerical的变量，求完他们之间的correlation，画一下correlation plot

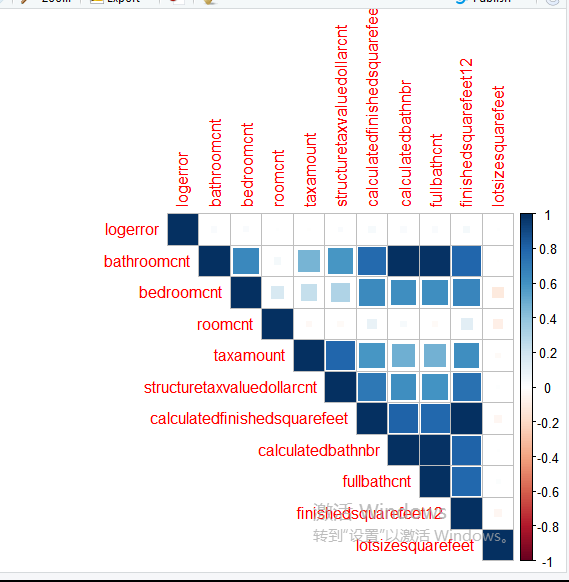
corrplot(correlations, method = "square", tl.cex = 1, type = 'upper')

# <http://www.sthda.com/english/wiki/visualize-correlation-matrix-using-correlogram>

tl.col (for text label color) and tl.srt (for text label string rotation) are used to change text colors and rotations.

tl.cex Numeric, for the size of text label (variable names).字符大小

根据画出的correlation plot找到和logerror相关性较强的variables，重点explore它们



安装“tabplot”包

package ‘fastmatch’ successfully unpacked and MD5 sums checked

package ‘ff’ successfully unpacked and MD5 sums checked

package ‘ffbase’ successfully unpacked and MD5 sums checked

package ‘tabplot’ successfully unpacked and MD5 sums checked

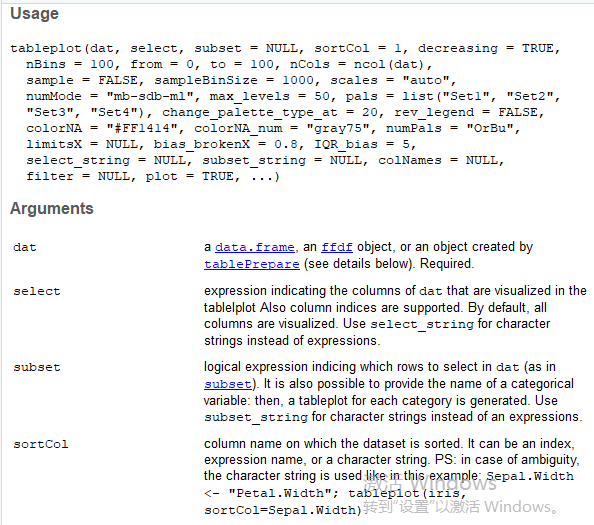
The downloaded binary packages are in

C:\Users\hp\AppData\Local\Temp\RtmpEvJWXY\downloaded\_packages

library(tabplot)

Tableplot——当你有很多variable在一个df中时，可以用这个plot一起explore一下，visualization看整体trend的方法

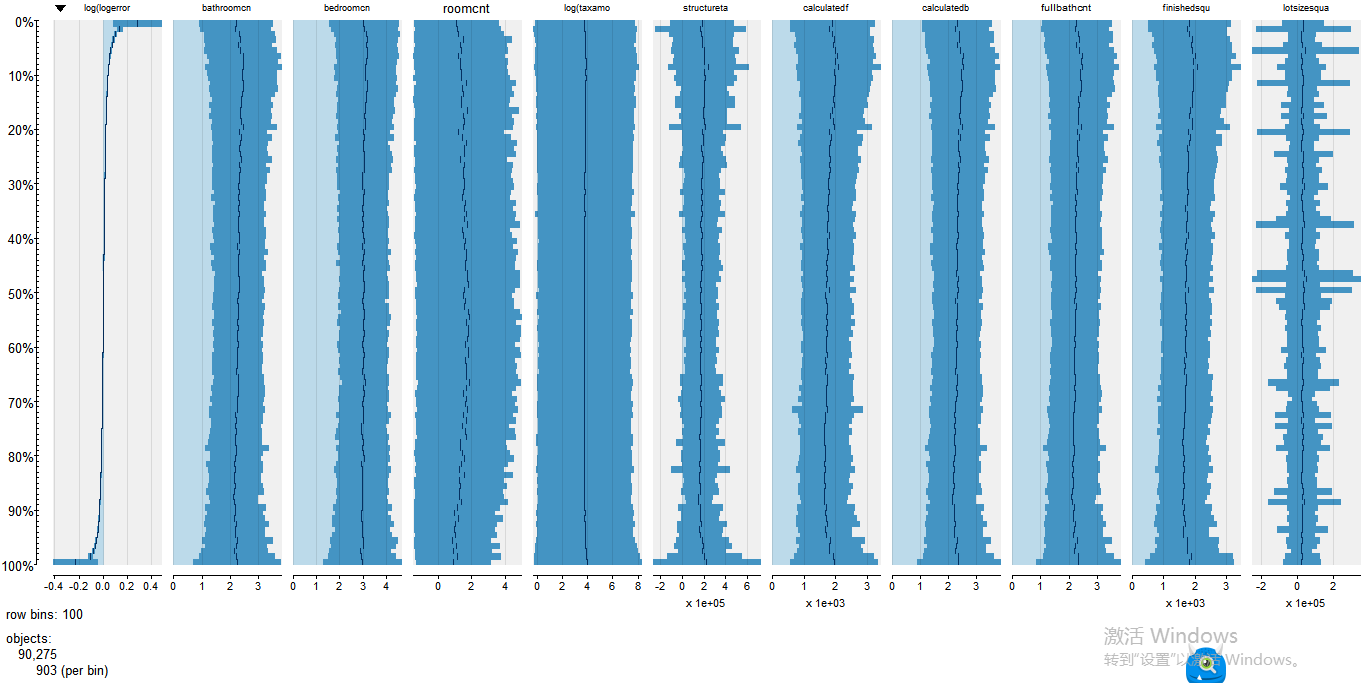
https://cran.r-project.org/web/packages/tabplot/vignettes/tabplot-vignette.html



tableplot(train, select = c('logerror', 'bathroomcnt', 'bedroomcnt', 'roomcnt',

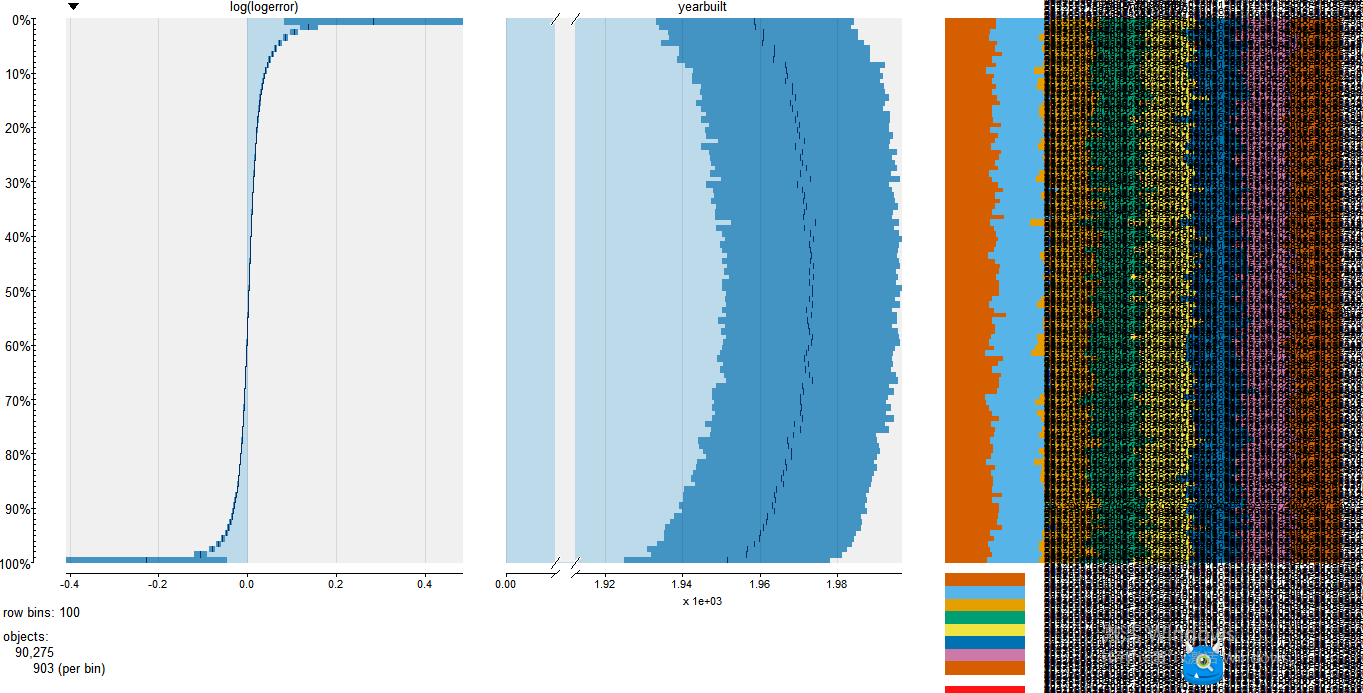
'taxamount', 'structuretaxvaluedollarcnt', 'calculatedfinishedsquarefeet',

'calculatedbathnbr', 'fullbathcnt', 'finishedsquarefeet12', 'lotsizesquarefeet'))



随着target variable logerror的变化，其他numerical变量的变化趋势

tableplot(train, select = c('logerror', 'yearbuilt', 'regionidcounty'))



**Data exploration**

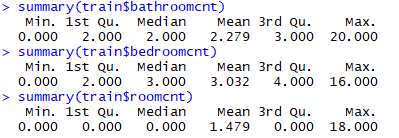
**First check each feature and features with relationship with each other**

# Take bathroomcnt, bedroomcnt and roomcnt for example,做proj要去check一些像这样的data

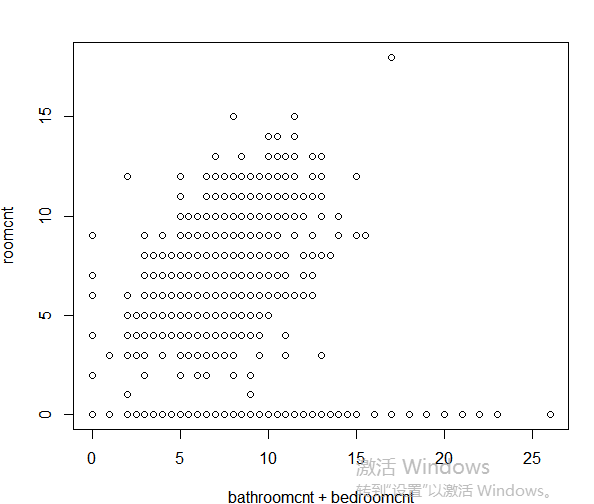
summary(train$bathroomcnt)

summary(train$bedroomcnt)

summary(train$roomcnt)



with(train, plot(bathroomcnt + bedroomcnt, roomcnt))



横坐标为浴室+卧室之和，纵坐标为总房间数，从这个图中能看出有问题——roomcnt为0（不是missing value，就是被记成了0）但bathroom+bedroom却不为0——data肯定有问题

有一些take home proj故意在data中埋一些点，有错误，看你能不能发现data的错误。

看这个你就要想到，可能有的roomcnt missing了，被记成了0.

还有一个问题就是roomcnt应该>= bathroom+bedroom，如果加一条对角线，所有的点应该在对角线上面。为什么下面还有这么多数据点？

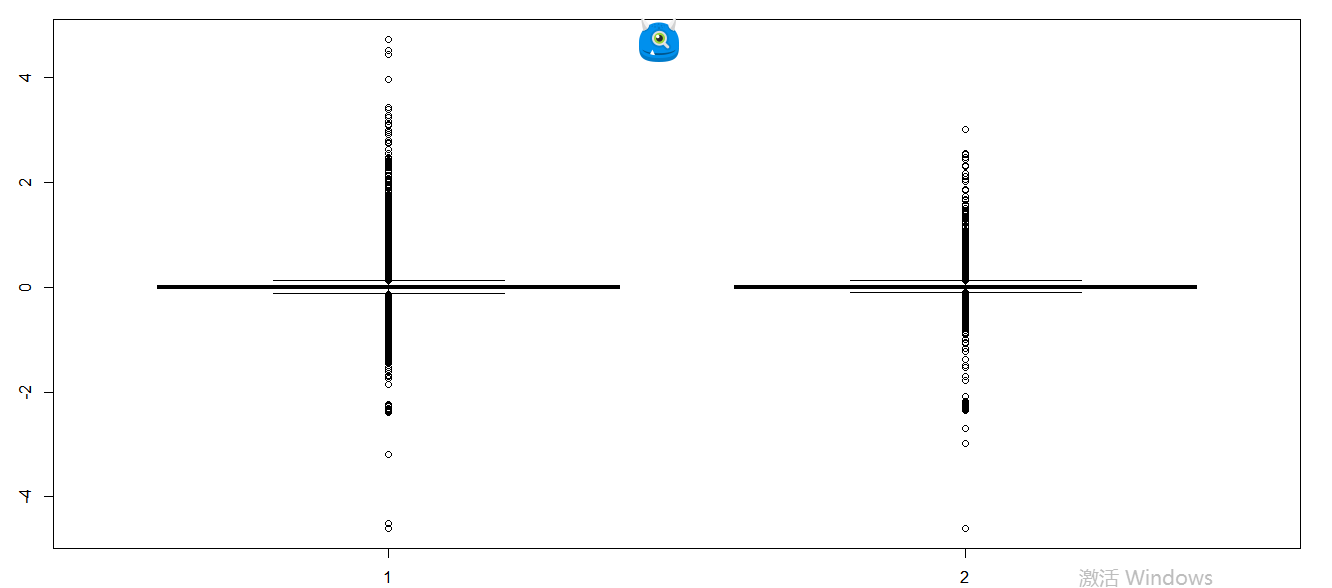
# Why there are so many house with roomcnt smaller than bath + bed?

# Assumption: error in data

那我们就画一下，两种情况：一种是roomcnt < bathroomcnt + bedroomcnt，画一下logerror的boxplot，另一种是roomcnt >= bathroomcnt + bedroomcnt，画一下logerror。

boxplot(subset(train, roomcnt < bathroomcnt + bedroomcnt)$logerror,

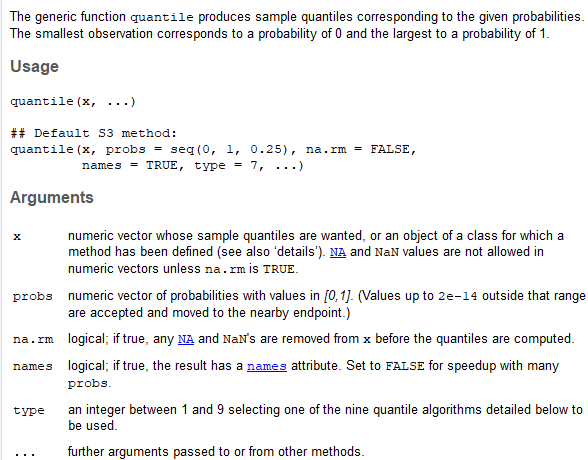
subset(train, roomcnt >= bathroomcnt + bedroomcnt)$logerror)



一样的问题，还是因为outliers看不太出来中间的细节，先找一下logerror的0.9 quantile是多少，来cut掉10%的outlier



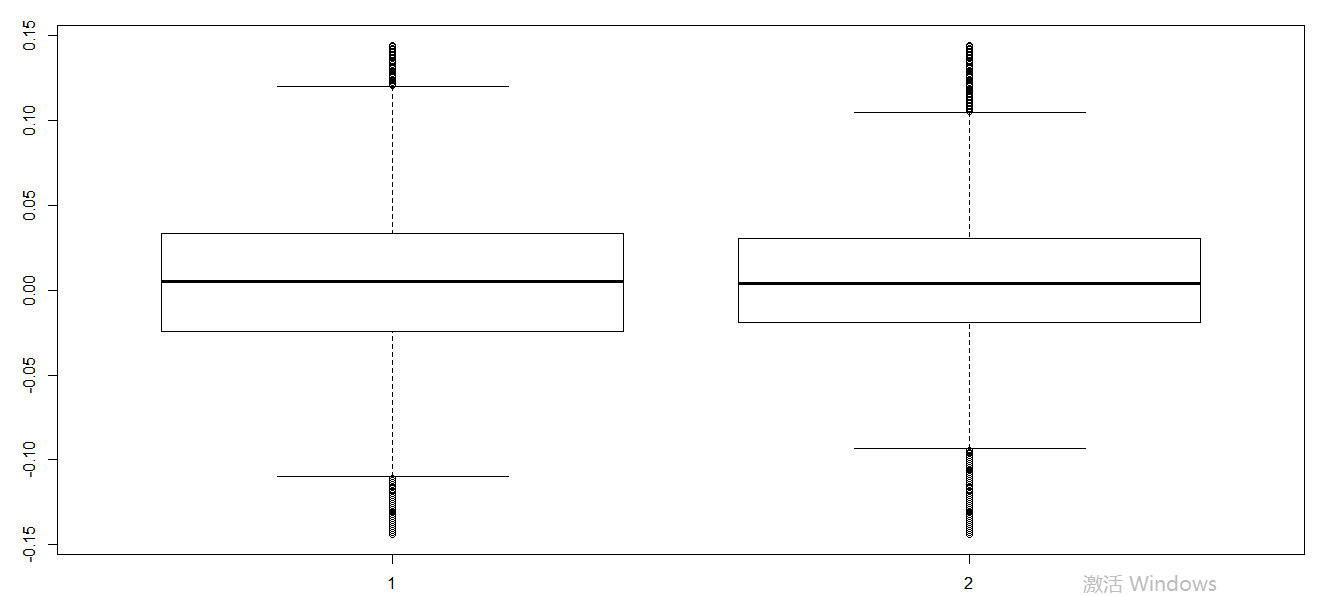
90%的logerror的绝对值都小于0.145，我只画90%以内的。





boxplot(subset(train, roomcnt < bathroomcnt + bedroomcnt **& abs(logerror) < 0.145**)$logerror,

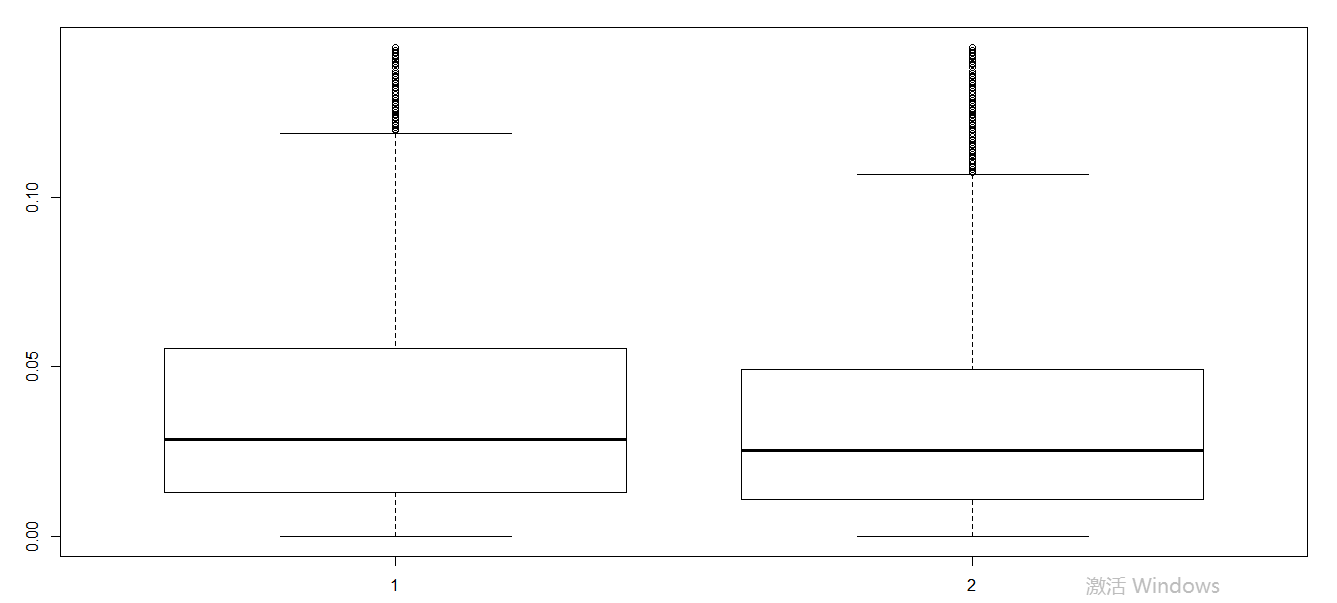
subset(train, roomcnt >= bathroomcnt + bedroomcnt **& abs(logerror) < 0.145**)$logerror)



看起来两组没有太大的区别。那再画一个图，看绝对值有没有区别？

boxplot(abs(subset(train, roomcnt < bathroomcnt + bedroomcnt & abs(logerror) < 0.145)$logerror),

abs(subset(train, roomcnt >= bathroomcnt + bedroomcnt & abs(logerror) < 0.145)$logerror))



看起来不太对的第一组的logerror的median高一点。我就想是不是data的错误导致了logerror不一样？

可以做一个T-test，看一下两组是否真的有差别？——在R中如何做T-tset——t.test

with(train, **t.test**(logerror ~ (roomcnt < bathroomcnt + bedroomcnt)))

With函数依然是说后续的操作都针对train这个dataset进行。T.test函数中 ~前的是y，是我们要看的target variable

~是指我们要针对哪两个组做test，~后是x，roomcnt < bathroomcnt + bedroomcnt是一个logical variable，它的可能取值只有T&F。

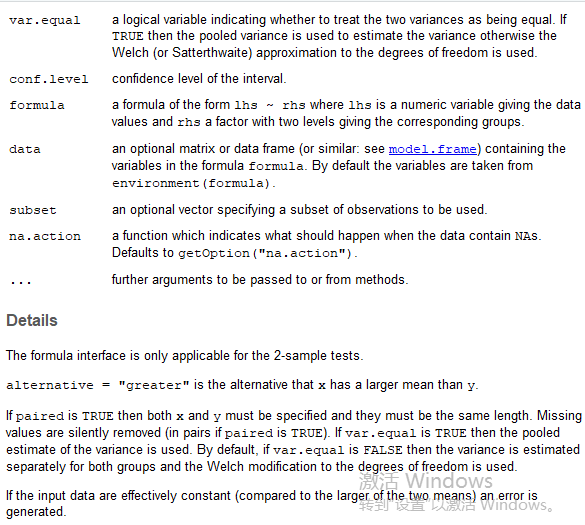
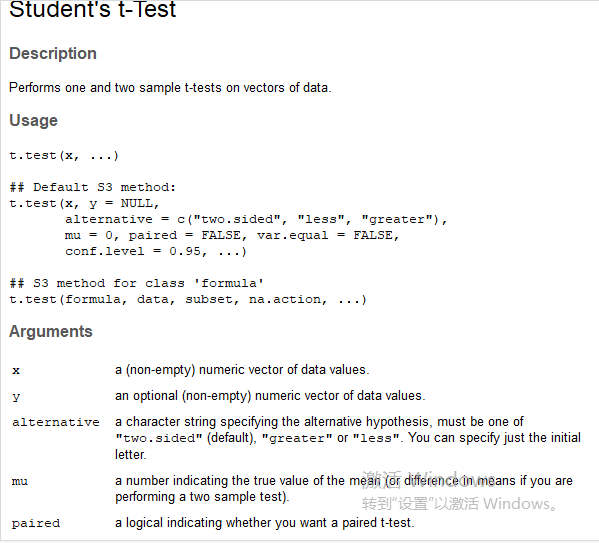
同样的写法：

train$room\_wrong = train$roomcnt < train$bathroomcnt + train$bedroomcnt

先在train中声明一个变量，命名为xxx,等于这个逻辑变量，将这个变量作为t-test中的x

t.test(tarin$logerror ~ train$room\_wrong)——最简单的t.test的syntax，就是y对于x的一个t.test。

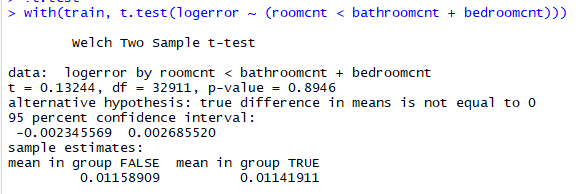




因为这两组的variance不一样所以就run了welch T test。

如果只放x一个参数，跑的就是One-sample t-test

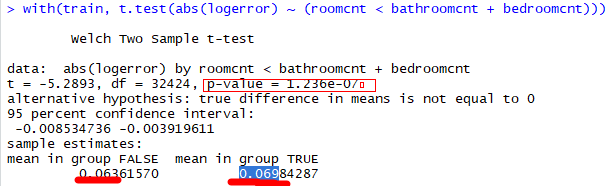
如果放一个x一个y，跑的是two-sample t-test



1. **value = 0.89**,非常不极端，**不能看出False组和True组的logerror mean的显著差别**。

刚刚根据箱型图也看到了，直接看logerror好像没什么区别，可以再run一个关于abs（logerror）的t.test：

with(train, t.test(abs(logerror) ~ (roomcnt < bathroomcnt + bedroomcnt)))



1. value非常小，说明这两个组确实是有区别的。

如何interpret这个结果？t.test(**tarin$logerror ~ train$room\_wrong**)

我现在的hypothesis就是——如果房间计数出错，predict出的logerror会比较大。

Confidence interval指的是Range of values that will happen 95% of the times.

**Cofidence interval**怎么看？这里的confidence interval是从负数到负数，**完全分布在负的这边**，**说明第二组明显比第一组要小**。

第一组的confidence interval**跨越了负数到正数**，0是在interval中间的，就说明**两个组并没有显著的差别**。（有时一样大，有时A比B小，有时A比B大，没啥差别）

一般看p-value就足够了，CI只是给你一个range，看一下margin是多大。

这个是很有可能的，比如本来5个房间，记成0，predict出的价格肯定会和实际售价的差别比较大，那logerror也会比较大。所以这个t-test就是想说“房间数错了的property”和“房间数没错的property”这两组之间的logerror是不是会有显著的差别。第一个t-test做的就是这个，结论是没有显著差别，因为p-value很大。但是进一步想，这个结果没有显著差别因为**logerror有正有负，而我们看的是均值**，**均值都是在0附近**，所以看不出差别来（很有可能是因为正负相互抵消造成的）。为了避免这种情况，我们改成看absolute logerror，看房间数记对记错**对logerror的绝对值**有无显著影响，这就是第二个t-tset。这个时候得出的p-value非常小，确实房间数记错了的时候logerror的绝对值会变大。

Conclusion/Insights: mean absolute value of log error is bigger for properties with wrong room count.

EDA部分回顾：我们做了什么——

1 先把data load 进来

2 对data整体做了简单的summary，初步understand

3 把两个表格的data combine起来，（subset property全集）因为我们只关心training data 的相关记录

4 处理missing values

5 做一些data exploation，比如看一些单独变量的statistics，画一些图（时间相关的&相关性），比如boxplot，分组的boxplots，correlation function, table plots。

6 看了一个关于房间数对目标变量的影响的一个例子。（只是一个例子，可以check很多不同的feature）

从对房间数的观察我们发现这个房间数可能有错，因为很多roomcnt比bath+bed小，我们因此做了assumption，来检查数据的错误对我们所关心的model的prediction结果（logerror）有无影响——针对这个问题我们画了两个boxplot，做了两个t-test。印证了假设。（数据采集的错误对zillow的price estimation结果确实有影响，会让这个estimation的误差变大）

发现了有影响，后续该怎么办？——我们现阶段只是做exploration，可以在最后的report中标注出来，说我发现我们的data有error，我们有没有可能fix 这个error来提高后续预测的准确度？

靠自己的脑洞，根据你探索数据发现的结论/insights来come up some suggestions。

Tableau是用来做快速的visualization，很难通过画图观察来做data exploration。很难做比如missing data imputation等的处理。R中可以。Tableau只能看，R可以做改变。