**求vector的长度**

length(a)

**求一个string的长度**

nchar(a)

截取vector部分

substr(a,1,7)

替代vector

gsub('behaviour','behavior',b)

合并 paste

paste(surname, givenname, sep = '/')

分割split

names <- 'Liu/Qizhang/Singaporan'

strsplit(names,'/')

**删去vector中的某一项**

f <- f[-1]

等差数列 seq

seq(from=5, to=20, by=3)

seq(5,20,length.out=10)

重复 rep

rep(1:5,3)

rep(1:5,each=3)

每个重复

rep(1:5,each=3)

扩量vector 直接加或者append

a <- c('12','bv')

b <- c('HG','JKK')

c <- c('BS','IU')

c(a,b,c)

b = append(a,'B')

vector里取一部分

a <- c(10,73,-19,1,0,78,90,65)

a[length(a):(length(a)-3)] *# R里面是可以反过来取值的*

vector命名字name

salaries <- c(James = 5000, Janice = 4588, Tom = 2598, Joanne = 7645, Jack = 6543)

名字找vector的value

salaries[c('Tom','James')]

找名字

names(salaries[salaries > mean(salaries)])

建list

Jack <- list(surname = 'Wong', givenname = 'Jack', age = 35, no.children=2, is.married = T, gender = 'male')

建matrix

m <- matrix(1:12, nrow=3, ncol=4)

m <- matrix(1:12, 3, 4, byrow=T)

建factor

company <- c(Jame = 'Engineer', Joanne = 'Manager', Tom = 'Clerk', Joe = 'Manager', Jason = 'Engineer', Janice = 'Clerk')

company <- as.factor(company)

company <- factor(company, levels = c('Engineer','Manager','Clerk'))

# lesson3

创建dataframe

names <- c('James','Jack','Joanne','Mike','Tom')

ages <- c(23,45,26,34,56)

salaries <- c(3000,5633,2984,9983,7356)

is.married <- c(T,F,F,T,T)

employees <- data.frame(names, ages, salaries, is.married)

求行数列数

length(employees)

## [1] 4

nrow(employees)

## [1] 5

dim(employees)

## [1] 5 4

给定行和列，建dataframe

c <- data.frame(matrix(NA,529,110))

改变整个dataframe的性质，变成factor看

police0 <- read.csv('Police Use of Force.csv',stringsAsFactors = T)

把character变成logical

police <- read.csv('Police Use of Force.csv')

police$OFFICER\_INJURY <- police$OFFICER\_INJURY=='Yes'

把整个dataframe变成numeric

b <- as.data.frame(sapply(b, as.numeric))

**把character变成时间**

police$INCIDENT\_DATE0 <- as.Date(police$INCIDENT\_DATE, '%m/%d/%Y')

*# Y是4位,后面的'%m/%d/%Y'是原始时间string 的储存方式*

police$INCIDENT\_DATE1 <- as.Date(police$INCIDENT\_DATE, '%m/%d/%y')

**求两个时间中间的时间**

cs$duration <- as.numeric(cs$deadline - cs$launched)

找满足条件的有多少个

sum(police$OFFICER\_YEARS\_ON\_FORCE < 5 & police$SUBJECT\_RACE == 'Black')

排序并且找前xx的的值

head(companies[order(-companies$Assets...billion.),'Company'],10)

#order出来的是行的序号

*#或*

companies[order(-companies$Assets...billion.),'Company'][1:10]

*#或*

**library**(dplyr)

companies %>% filter(Continent == 'Asia' & companies$Assets...billion. > 50) %>% mutate(Profit.Margin = Profits...billion./Sales...billion.) %>% arrange(-Profit.Margin) %>% top\_n(10) %>% select(Company)

#arrange也可以排序，*这个输出的结果还是一个dataframe，只不过是按照pm列从小到大排好了的，加了-就是从大到小了*

pivot table数据透视表

companies %>% group\_by(Continent) %>% summarise(Avg.Asset = mean(Assets...billion.))

companies %>% group\_by(Continent, Country) %>% summarise(Avg.Asset = mean(Assets...billion.) , number = n())

如果两个指标划分，但只输出一列的话，避免warning可以

df <- p %>% group\_by(SUBJECT\_RACE, TIME.SCENARIO) %>% summarise(number\_of\_cases=n(),.groups = 'drop')

变成tidy/neat format/ long format

**library**(dplyr)

**library**(tidyr)

home\_l <- home\_w %>% gather(Year,Rate,-Property.Type) *# 所有列除了Property.Type*

ed1 <- ed %>% gather(Year,Count,2:26)

把long format变成wide format 用到spread()

house\_w2 <- home\_l %>% spread(Year,Rate)

增加新列

No.Of.Childen <- c(1,2,0,2)

a <- cbind(a,No.Of.Childen)

增加新行 用rbind函数

Rose <- data.frame(names = 'Rose', position = 'Clerk', Date.Of.Join = '2021-01-02', No.Of.Childen = 0 )

a <- rbind(a, Rose)

增加新行，但有数据缺失 用bind\_rows

zzx <- data.frame(names = 'zzx', position = 'student')

a <- bind\_rows(a,zzx)

合并

*#1.inner join两个table中查找值相同的 行*

set1 <- merge(companies, countries, by = 'Country')

*#2.outer join 返回两个table里所有的行，有找不到的数据就会标成na*

set2 <- merge(companies, countries, by = 'Country', all = T)

*#3.left join 返回第一个table里所有的行，找不到的数据就会标成na*

set3 <- merge(companies, countries, by = 'Country', all.x = T)

*#4.right join 返回第二个table里所有的行，找不到的数据就会标成na*

set4 <- merge(companies, countries, by = 'Country', all.y = T)

# lesson4

读取tsv，每个column用tab隔开

data <- read.table("custdata.tsv",header=T,sep="\t")

读取csv，表示用comma隔开的数据

data <- read.table("Countries.csv",header=T,sep=",")

读取xml

<>叫做tag

在<>中的叫做attribute

有一对 <>和</> 在这里对中间的东西叫做text/value

**library**(XML)

data <- xmlParse("books.xml")

**l** <- xmlRoot(data)

children.lib <- xmlChildren(**l**)

找value

*#1 这个方法只能找1个value的*

xmlValue(children.lib[1])

*#2 可以找很多value的 继续用xmlChildren() + xmlToDataFrame()*

*#xmlChildren()是把node变成list*

books <- xmlChildren(children.lib[[2]]) *#[[]]变成node了*

*#xmlToDate是把list变成dataframe*

books <- xmlToDataFrame(books)

也可以直接用path得到值

data <- xmlParse("books.xml")

*#getNodeSet得到的也是list*

books <- getNodeSet(data,"/library/catalog/book[@type='HardCover']")

books <- xmlToDataFrame(books)

读取jsonlite数据

**library**(jsonlite)

url <- "https://api.data.gov.sg/v1/transport/carpark-availability"

data <- fromJSON(url)

carpark <- data$items$carpark\_data[[1]]

先读几行判断类型

colclass <- c("character","character","character","character","character","character","factor","factor","factor","character","factor","factor","factor")

hospital <- read.csv("hospital-data.csv",colClasses=colclass)

# lesson5

柱状图

**library**(ggplot2)

ggplot(data = countries, aes(x=Continent)) + geom\_bar(stat='Count', fill='pink', width=0.5, color='white')

有x有y轴

country.count <- countries %>% group\_by(Continent) %>% summarise(Count = n()) %>% as.data.frame()

ggplot(data = country.count, aes(x=Continent, y=Count)) + geom\_bar(stat='identity', fill='pink', width=0.5, color='white')

排序

country.count <- country.count %>% arrange(Count)

country.count$Continent <- factor(country.count$Continent, level = country.count$Continent)

ggplot(data = country.count, aes(x=Continent, y=Count)) + geom\_bar(stat='identity', fill='pink', width=0.5, color='white')

或者

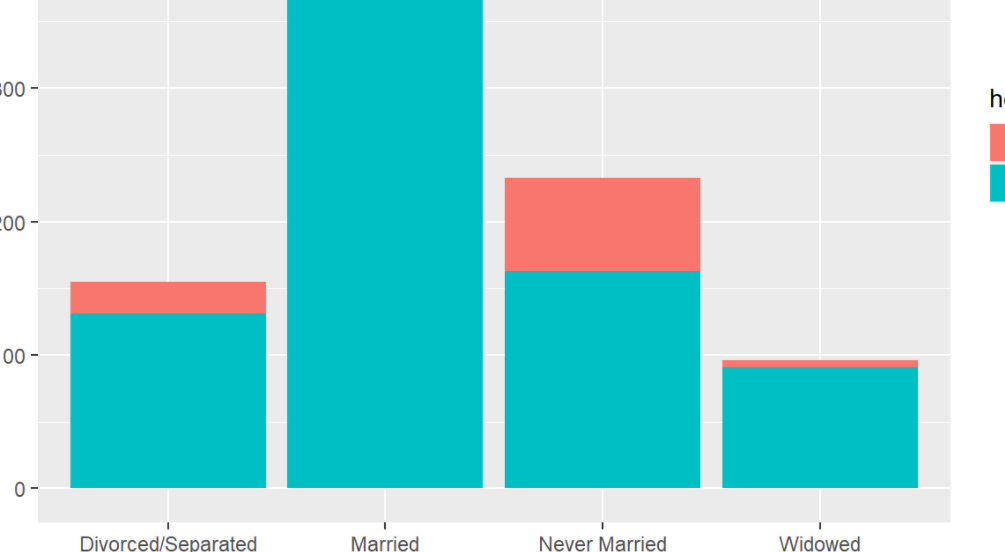
ggplot(data=cs,aes(x=reorder(main\_category,ppb),y=ppb) 把main\_category按ppb排序

**涉及多个比较对象**

比较不同婚姻状况的人中，买保险的数量(涉及了两列的数据)

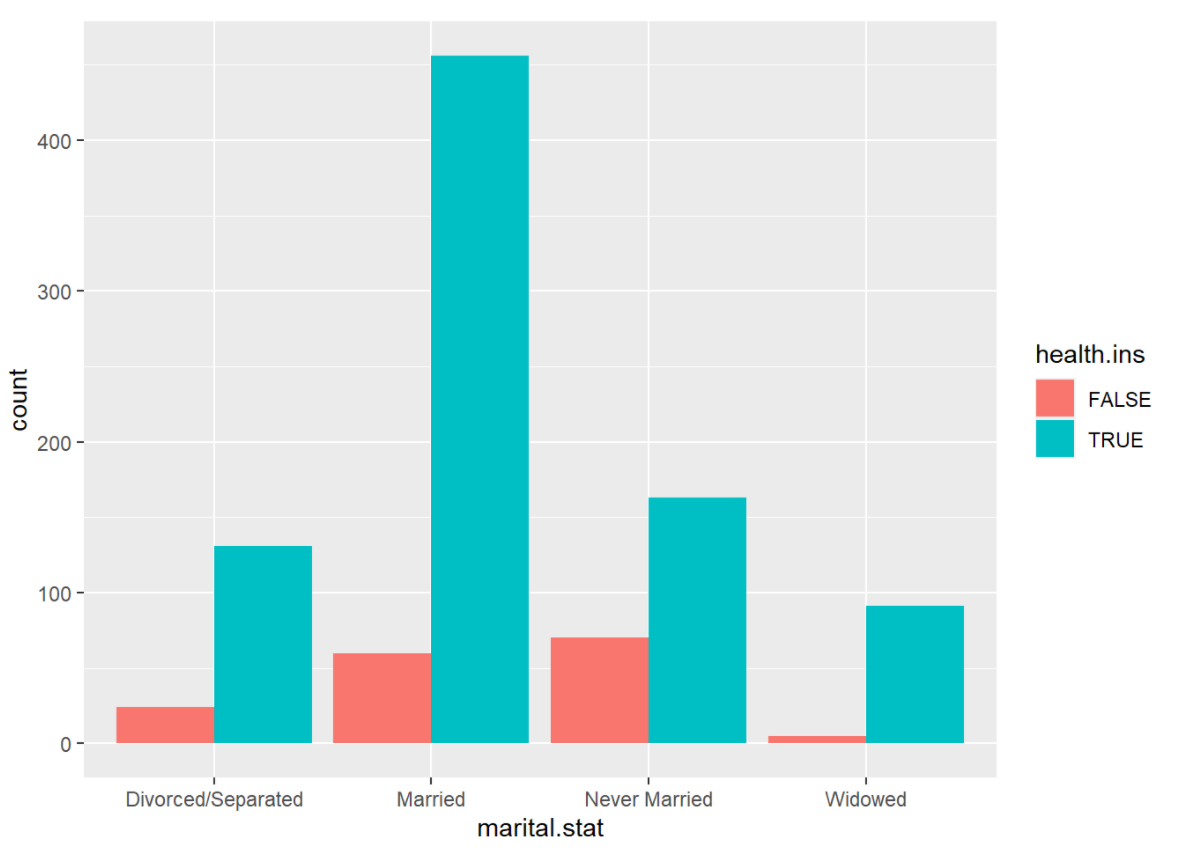
*#普通的bar两个会在一列上面*

ggplot(custdata) + geom\_bar(aes(x=marital.stat, fill=health.ins))



在一列上面看的不清晰，可以使用dodge变成两列

ggplot(custdata) + geom\_bar(aes(x=marital.stat, fill=health.ins), position='dodge')



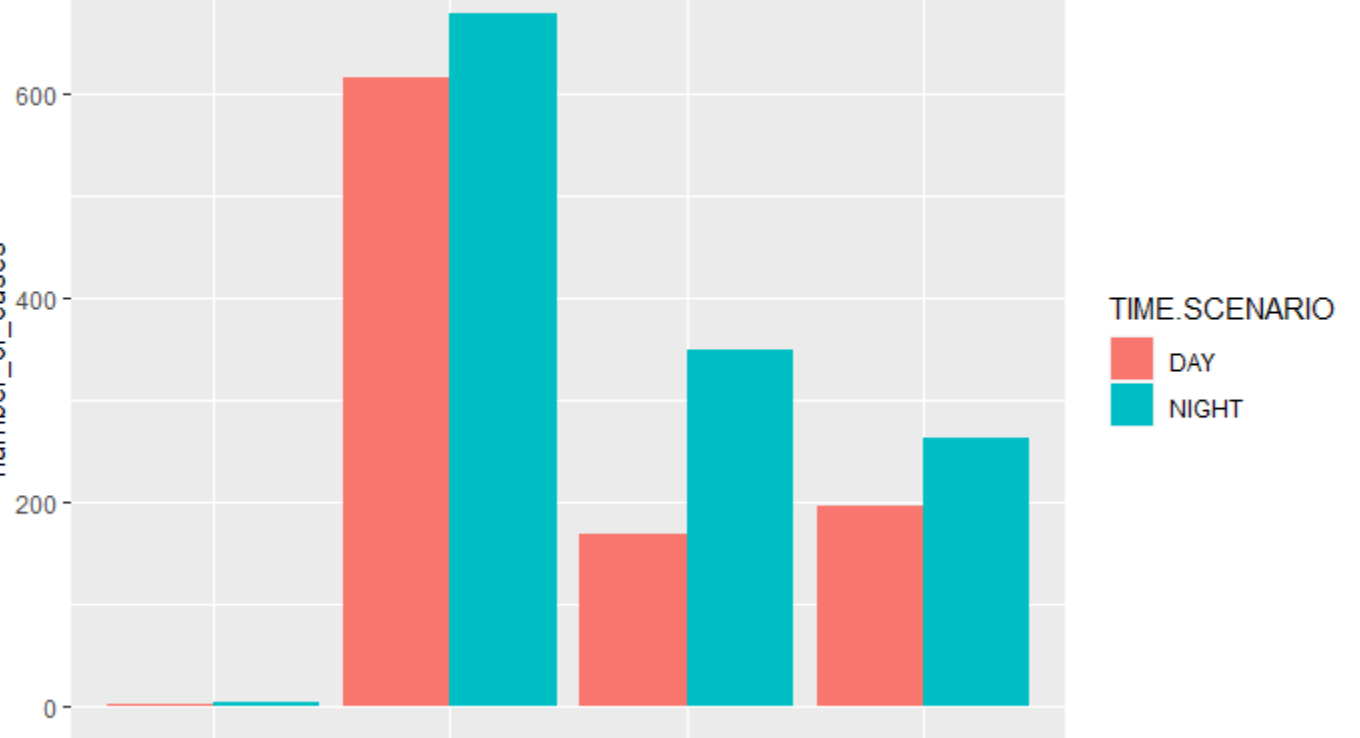
看占比fill

ggplot(custdata) + geom\_bar(aes(x=marital.stat, fill=health.ins), position='fill')

*#得出的结论是：widow has the highest tendency of buying insurance product.*

用两个指标来划分，并且展示每个有多少

ggplot(data=df,aes(x=SUBJECT\_RACE,y=number\_of\_cases,fill=TIME.SCENARIO)) + geom\_bar(stat='identity', position='dodge')



折线图

**library**(ggplot2)

ggplot(data = country.count, aes(x=Continent, y=Count)) + geom\_line(group=1, color='black', linetype=2) *#group=1指名字是1*

点图

ggplot(data = country.count, aes(x=Continent, y=Count)) + geom\_point(size=4, color='black', shape=19) + xlab(’’) + ylab(’’)

几种图layer重合

ggplot(data = country.count, aes(x=Continent, y=Count)) + geom\_point(size=4, color='blue', shape=25) + geom\_line(group=1, color='pink', linetype=4) *#哪个后写的哪个在上面*

不同数据集source里的

ggplot() + geom\_bar(data=countries,aes(x=Continent),fill='pink') + geom\_point(data=country.count, aes(x=Continent,y=Count), size=2, color='blue', shape=25) + geom\_line(data=country.count, aes(x=Continent,y=Count), group=1, color='green', linetype=4)

根据某一数据上color

ggplot(countries, aes(x=GDP\_Per\_Capita, y=Life\_Expectancy,size=Population, color=Continent)) + geom\_point()

probability/density of each data

ggplot(countries,aes(x=GDP\_Per\_Capita)) + geom\_density()

把lognormal变成normal

ggplot(countries,aes(x=log10(GDP\_Per\_Capita))) + geom\_density()

就可以把none liner relationship不好，所以要通过log变成liner relationship

g1 <- ggplot(countries, aes(x=GDP\_Per\_Capita, y=Life\_Expectancy, size=Population, color=Continent)) + geom\_point() + scale\_x\_log10()

*#譬如得到y=10.4\*log10(x)+25*

*#log10(x)增加1，y增加10.4；也就是x增加10倍，y增加10.4。 x increases by 10 times, y increases by 10.4。*

Scale the color这个是针对前面的color=Continent

g1 + scale\_color\_manual(values=c('red','green','yellow','black','blue','orange'))

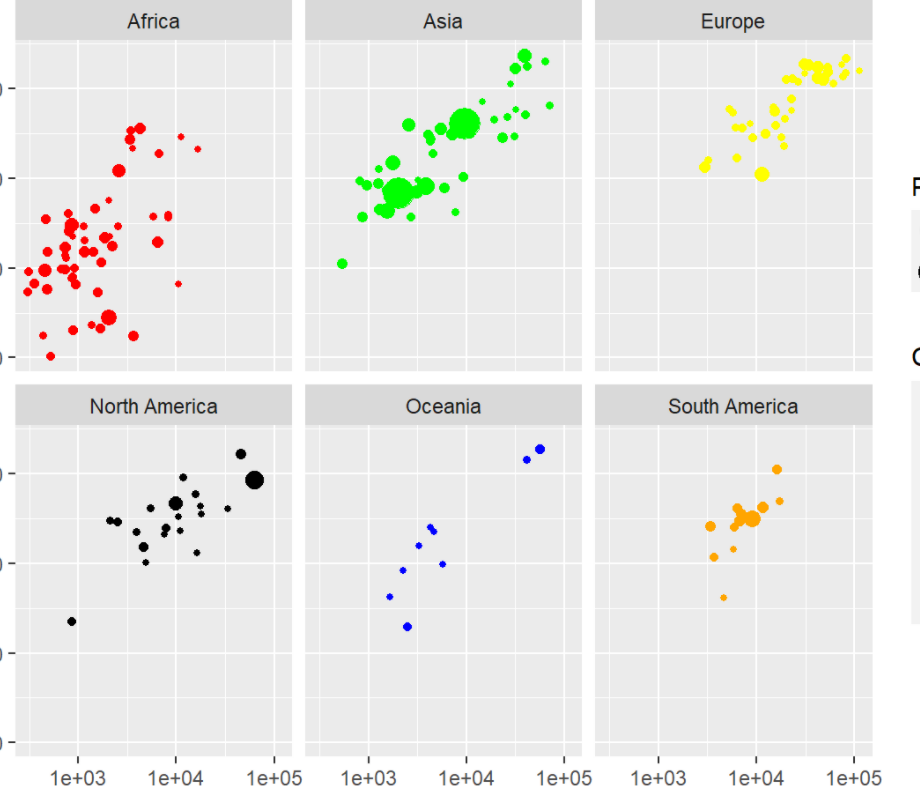
polar system

g2 <- ggplot(data = countries, aes(x=Continent, fill = Continent)) + geom\_bar(width=0.5)

g2 + coord\_polar()

faceting 分层

g1 + scale\_color\_manual(values=c('red','green','yellow','black','blue','orange')) + facet\_wrap(~Continent)*#也可以ncol=1*



横着放一行

g1 + scale\_color\_manual(values = c('red','green','yellow','black','blue','orange')) + facet\_wrap(~Continent, nrow=1)

plotly package可以做interactive one

**library**(plotly)

ggplotly(g1)

gganimation package动画

**library**(gganimate)

**library**(gapminder) *#包含很多国家数据的数据库*

g4 <- ggplot(gapminder, aes(x=gdpPercap, y=lifeExp, size=pop, color=continent)) + geom\_point() + scale\_x\_log10() + theme\_bw()

g <- g4 + transition\_time(year) + labs(title='Year: {frame\_time}')

animate(g,renderer=gifski\_renderer())

**做animation**

g1 <- ggplot(vax\_countries, aes(x = GDP, y = total\_vaccinations\_log10)) +

geom\_point() +

scale\_x\_log10() + #像这里有时候数字太大了，可以log10一下，同时y也可以

xlab("G.D.P.") +

ylab("# of Total Vaccinations")

g1 <- g1 + transition\_time(date) + labs(title = "{frame\_time}")

animate(g1, renderer = gifski\_renderer())

jitter 防止重合

ggplot(custdata,aes(x=marital.stat,y=age)) + geom\_point() + geom\_jitter(position=position\_jitter(0.2)) + ylim(c(0,100))

boxplot

*#color=health.ins用颜色分辨有没有买保险*

ggplot(custdata,aes(x=marital.stat,y=age,color=health.ins)) + geom\_boxplot(color='black') + geom\_jitter(position = position\_jitter(0.2)) + ylim(c(0,100))

*#boxplot的颜色按照前面ggplot的color决定*

ggplot(custdata,aes(x=marital.stat,y=age,color=health.ins)) + geom\_boxplot() + geom\_jitter(position = position\_jitter(0.1)) + ylim(c(0,100))

# Lesson6

改大标题

标签 labs(title=, subtitle=)

改变x、y轴标题

+xlab('Launched date')

+ylab(‘’)

改标题的格式

+theme(axis.title.x = element\_text(size=12, color='blue',face='italic')

改xy轴指标的格式竖着

+ theme(axis.text.x = element\_text(angle=90))

改指标的名称**scale\_x\_discrete(label=)**

*#因为这里的value要么是true，要么是false*

hasinsurance <- **function**(x)

{

**return**(ifelse(x,'Has insurance','No insurance'))

}

ggplot(custdata,aes(x=health.ins,fill=health.ins)) + geom\_bar(width=0.3) + **scale\_x\_discrete(label=**hasinsurance)

改背景

+theme(panel.background=element\_rect(fill='lightgrey'))

Ggmap

有不同的map type： satellite, hybrid, toner, water, color, terrain-background, toner-lite

**library**(ggmap)

SG <- get\_map('Singapore',zoom=11, maptype = 'satellite')

ggmap(SG)

把图和已有的数据合并起来

ggmap(SG) + geom\_point(data=pizzahut, aes(x=Lon, y=Lat), size=3, color='red')

调整透明度来显现density

sfmap <- get\_map('San Francisoo', zoom=12, maptype='toner-lite')

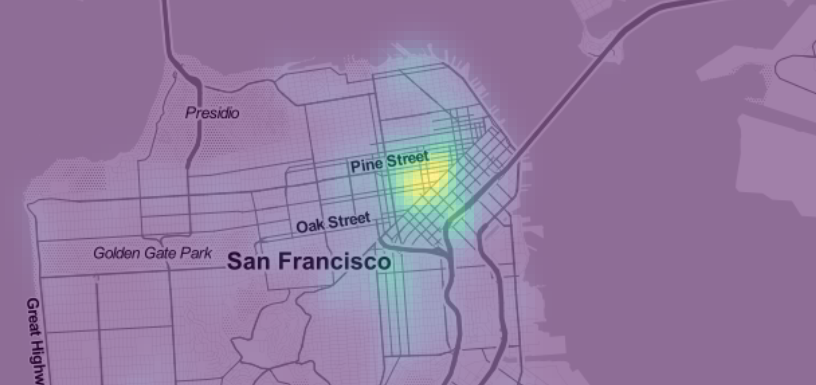
ggmap(sfmap) + geom\_point(data=sf, aes(x=lon,y=lat), alpha=0.05)

**heatmap来展现 stat\_density2d(fill=..density..)**

**library**(viridis)*#这是一个调色的安装包*

ggmap(sfmap) + stat\_density2d(data=sf, aes(x=lon,y=lat, fill=..density..),geom='tile',contour = F,alpha=0.5) + scale\_fill\_viridis(option='inferno')

# contour等高线



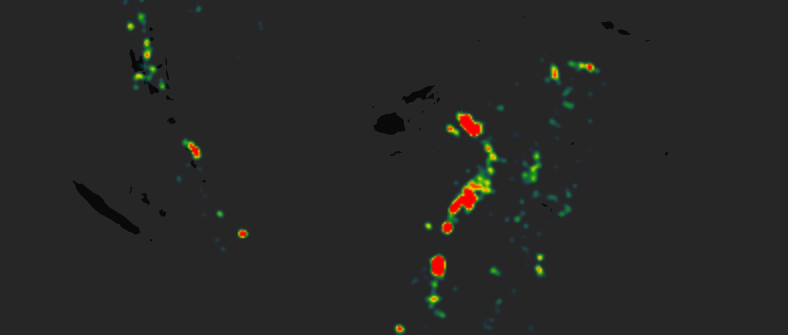
或者用leaflet

**library**(leaflet)

**library**(tidyverse)

**library**(leaflet.extras)

m <- leaflet(quakes) %>% addProviderTiles(providers$CartoDB.DarkMatter) %>% **addWebGLHeatmap**(lng=~long,lat=~lat,size=60000)*#each event colored has a physical radius of 6000 metres*



based on magnitudes

leaflet(quakes) %>% addProviderTiles(providers$CartoDB.DarkMatter) %>% addWebGLHeatmap(lng=~long,lat=~lat,**intensity=~mag**,size=10000)

m <- leaflet(p) %>% addProviderTiles(providers$CartoDB.DarkMatter) %>% addWebGLHeatmap(lng=~LOCATION\_LONGITUDE,lat=~LOCATION\_LATITUDE,size=600) %>% setView(lat = 32.78220, lng = -96.79746,zoom = 11)

**边界borderborder polygon coloring 按照数字划分颜色**

先找一个dataframe，里面的column是name和polygon；

再找一个dataframe，里面的column是name和population。

把d2和d1用name合并起来

**library**(raster)

**library**(rgdal)

**library**(XML)

*#gadm是一个地图的database*

SG <- getData('GADM', country='SG', level=1)

*#看看SG里面包含了什么 用@*

SG@data

*#可以用于查看是怎么划分区域的*

SG$NAME\_1

*#直接加了一列*

SG$pop <- c(1928932,3874774,766666,2883434,7376362)

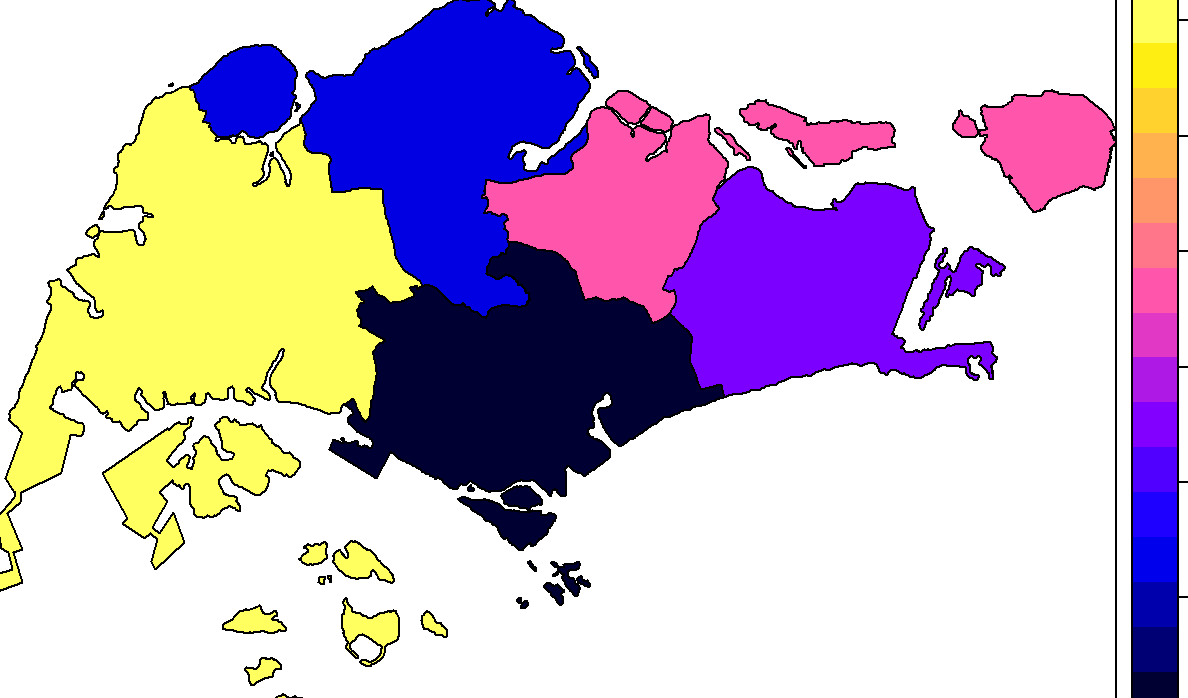
spplot(SG,'pop')

用merge把两个dataframe中的数量和地图合并起来

population <- data.frame(Region=c("Central","East","North","North-East","West"),value=c(23435,43451,34254,56432,78131))

SG@data <- merge([SG@data,population,by.x='NAME\_1',by.y='Region](mailto:SG@data,population,by.x='NAME_1',by.y='Region)')

spplot(SG,'value')



用leaflet

**library**(raster)

**library**(leaflet)

*#前面和spplot的操作一样*

d1 <- data.frame(Region=c('Central','East','North','North-East','West'),Value=c(3,4,1,7,10))

SG <- getData('GADM', country='SG', level=1)

SG@data <- merge(SG@data,d1,by.x='NAME\_1',by.y='Region')

*#这可以搞个标签*

popup <- paste0("<strong>Name: </strong>",SG$NAME\_1)

*#创建一个调色板palette 按照降水量来显示颜色 降水量越大 颜色越深*

pal <- colorNumeric(palette='Blues', *#Blues一系列的蓝色*

domain=SG$Value) *#通过给定的paltte，将数据映射到颜色，数字越大颜色越深*

m <- leaflet() %>% addTiles() %>% addPolygons(data=SG,

weight=2,*#描边宽度*

stroke = TRUE,

smoothFactor = 0.1, *#缩放简化折线的程度(越大性能好清晰度差)*

fillOpacity = 0.8, *#填充不透明度*

color = ~pal(Value),

popup=popup)

随便给的颜色

factpal <- colorFactor(topo.colors(5),SG$Value)

m <- leaflet() %>% addTiles() %>% **addPolygons**(data=SG,weight=2,stroke=F,smoothFactor = 0.2,fillOpacity = 0.8,color=~factpal(Value),popup = popup)

**把国家命名标准化，用countrycode这个包。一个是genc3c，一个是iso code**

library(countrycode)

country$cd <- countrycode(country$Country, origin = "country.name", destination = "genc3c")

countries$iso\_code <- countrycode(countries$Country, origin = 'country.name', destination = 'iso3c')

**把小数舍去round**

data$cfp <- round(data$Confirmed / data$Population \* 100000)

**转为浮点数，便于相乘**

v$Population <- as.double(v$Population)

**加一条线**

a <- mean(cs$ppb)

ggplot(data=cs,aes(x=reorder(main\_category,ppb),y=ppb)) + geom\_bar(stat = 'summary', fun.y = 'mean') + theme(axis.text.x=element\_text(angle=90)) + **geom\_hline(yintercept=a,colour = 'blue', size = 0.7)** + labs(y='Average amount pledged per backer', x='Category')

**控制y轴的范围**

+ coord\_cartesian(ylim = c(0,60000))

**Aids the eye in seeing patterns in the presence of overplotting帮助眼睛在过度绘制的情况下看到图案**

ggplot(data=cs,aes(x=duration, y=usd\_pledged\_real)) + geom\_point(alpha=0.1) + geom\_smooth(color='blue') + scale\_y\_log10()

**把date变成年月的形式**

format(as.Date(df1$launched),"%Y-%m")

再加上每个月的第一天

df1$launched <- as.Date(paste0(**format(**as.Date(df1$launched), "%Y-%m"**)**,'-01'))

**变成时间的形式**

library(data.table)

p <- p %>% mutate(intimenew = as.ITime(as.POSIXct(INCIDENT\_TIME, format = '%I:%M:%S %p')))

a <- as.ITime(as.POSIXct('6:00 AM', format = '%I:%M %p'))

b <- as.ITime(as.POSIXct('7:00 PM', format = '%I:%M %p'))

**按x轴的值对y轴上的变量进行计算**

+geom\_bar(stat = 'summary', fun.y = 'mean')

+geom\_line(aes(x=launched,y=usd\_pledged\_real),stat='summary',fun.y='median')

**上色color**

色板colormap

library(RColorBrewer)

cols <- colorRampPalette(brewer.pal(9, 'YlOrRd'))(length(countries$GDP\_Per\_Capita))

**删除dataframe中的某列或者保留某列**

p <- subset(p, select = -INCIDENT\_TIME\_new)

d$residual <- NULL

**筛选不符合条件的某列**

cs <- cs[!(cs$backers==0 & cs$state=='successful'),]

**删除某行**

ed <- ed[c(-1,-17),]

#或者

gdp\_map <- subset(gdp\_map, continent != "Antarctica")

**删除na的行**

na.omit(your.data.frame)

‘

**删除某个变量**

**rm(a)**

**combined bar and line with legend堆积图**

ggplot(df1,aes(x = time))+

geom\_bar(aes(y = number,fill = is\_canceled),stat="identity") +

geom\_line(aes(y=adr\*50, color=''),stat="identity", group = 1) +

geom\_point(aes(y=adr\*50)) +

scale\_y\_continuous("Number of records", sec.axis = sec\_axis(~ . /50, name = "Price")) +

scale\_color\_manual('', labels = 'Price', values = 'black') +

xlab('Arrival time') +

labs(title='Distribution of records and price over the time')

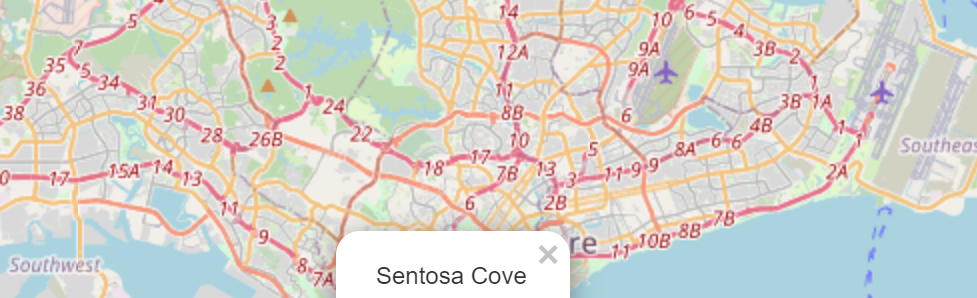
**Leaflet**

一个

**library**(leaflet)

library(tidyverse)

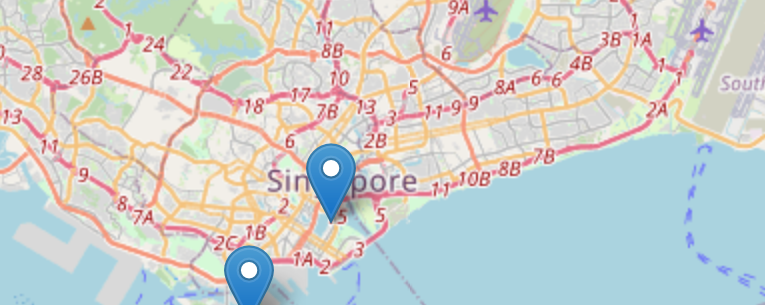
my\_map <- leaflet() %>% addTiles () %>% addMarkers(lat = 1.25011 , lng = 103.83093 , popup ="Sentosa Island")



多个

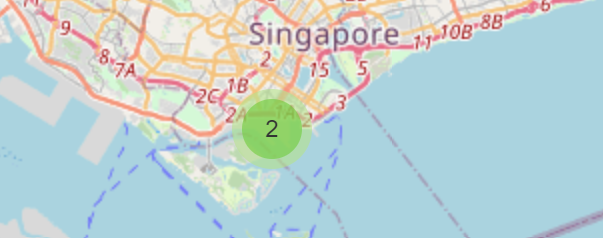
df <- data.frame(lat = c(1.25011,1.28544, 1.318707, 1.282375, 1.4043, 1.4022, 1.4029 , 1.2868 , 1.3332, 1.2893 ) , long =c(103.83093 ,103.859590, 103.706442 , 103.864273 , 103.7930 , 103.7881 , 103.7917, 103.8545, 103.7362 , 103.8631) , popup = c("Sentosa Island", "Marina Bay Sands " , "Jurong Bird Park","Gardens By the Bay", "Singapore Zoo" , "Night Safari", "River Safari", "Merlion Park","Science Center", "Singapore Flyer") )

my\_map <- leaflet() %>% addTiles() %>% addMarkers(data = df , lng = ~long , lat = ~lat , popup = df$popup)



**cluster the attractions 圈出来一块区域**

my\_map %>% leaflet() %>% addTiles() %>% addCircleMarkers(data = df , lng = ~long , lat = ~lat , radius = 5 , clusterOptions = markerClusterOptions())



We can make the map more expressive by adding Icons

sgIcon <- leaflet::makeIcon(iconUrl = "http://img.freeflagicons.com/thumb/speech\_bubble\_icon/singapore/singapore\_640.png", iconWidth = 41\*225/240 , iconHeight = 41, iconAnchorX = 31\*215/230/2, iconAnchorY = 16 )

my\_map <- leaflet() %>% addTiles() %>% addMarkers(data = df , lng = ~long , lat = ~lat , popup = df$popup , icon = sgIcon)

**做一个能选择的区域和地图模式的interactive的map**

pizzahut.location <- read.csv('PizzaHut.csv')

*#先划分4个层*

region.list <- c('North','East','Central','West')

*#每个region给不同的颜色*

colorFactors <- colorFactor(c('red','green','blue','brown'),domain=pizzahut.location$Region)

m <- leaflet() %>% addTiles()

*#把所有数据分为4层，对应上面的region，每一层建成一个dataframe*

**for**(i **in** 1:4)

{

pizzahut.region <- pizzahut.location[pizzahut.location$Region == region.list[i],]

m <- addCircleMarkers(m,

lng=pizzahut.region$Lon,

lat=pizzahut.region$Lat,

popup = pizzahut.region$Address,

radius=10,

stroke = F,

fillOpacity=1,

color = colorFactors(pizzahut.region$Region),

group = region.list[i])

}

*#在前面不同区域图层的基础上增加不同map的形式*

*#http://leaflet-extras.github.io/leaflet-providers/preview/index.html*

*#参考这个网站里面有不同的provider*

m <- addTiles(m,group='Default')

m <- addProviderTiles(m,'Esri.WorldImagery',group='Esri')

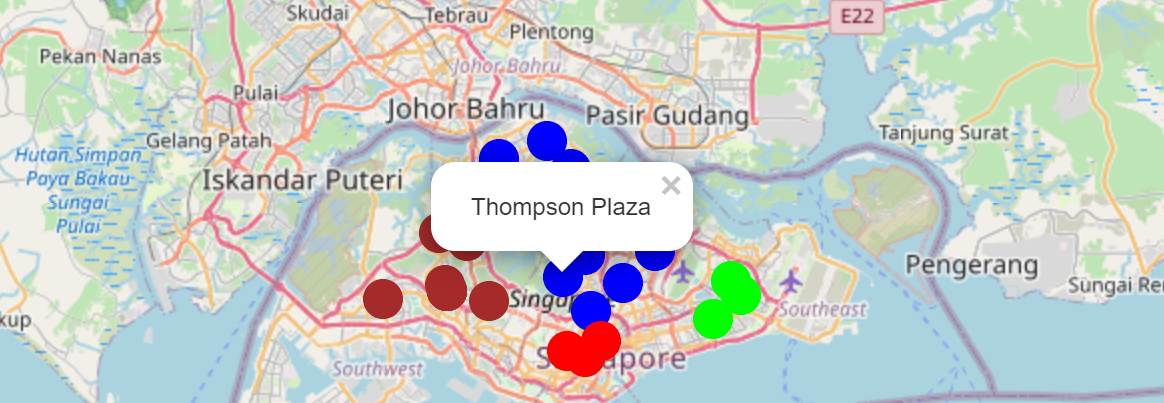
m <- addProviderTiles(m,'Stamen.Toner',group='Toner')

m <- addProviderTiles(m,'Stamen.TonerLite',group='Toner Lite')

m <- addLayersControl(m,

baseGroups = c('Default','Esri','Toner','Toner Lite'),*#这个是每次只能出现一个*

overlayGroups = region.list)*#overlaygroups是可以共存的 coexist的group*



**随机生成一组序列**

rnorm(mean=100,sd=1,200)

**判断在不在一组vector里面**

r <- c('White', 'Black', 'Hispanic','Asian')

p <- p %>% filter(SUBJECT\_RACE %in% r)

**数用comma隔开的数据有多少个**

p$cuof <- count.fields(textConnection(p$UOF\_NUMBER), sep = ",")

**算用comma隔开的element的总value是多少**

p$N\_new<- sapply(strsplit(p$NUMBER\_EC\_CYCLES, ", "), function(x) sum(as.numeric(x)))

**character转化成货币形式**

sales <- c(1736134, 10034,1003948,209445,98878.76,398454,777734.12,1039489.34,293894,9834,9384754.65)

s1 <- paste0("$", formatC(as.numeric(sales), format="f", big.mark=","))

**01，02………**

number <- formatC(1:length(sales),width = 2, flag = 0)

**strsplit的使用**

s3

[1] "Athens, Paris, St. Louis, London, Stockholm, Berlin , Antwerp, Paris, Amsterdam, Los Angeles, Berlin, Tokyo , London , London, Helsinki, Melbourne, Rome, Tokyo, Mexico City, Munich, Montreal, Moscow, Los Angeles, Seoul, Barcelona, Atlanta, Sydney, Athens, Beijing, London, Rio de Janeiro, Tokyo"

strsplit(s3,', ')变成一个列表

[[1]]

[1] "Athens" "Paris" "St. Louis" "London" "Stockholm"

[6] "Berlin " "Antwerp" "Paris" "Amsterdam" "Los Angeles"

[11] "Berlin" "Tokyo " "London " "London" "Helsinki"

[16] "Melbourne" "Rome" "Tokyo" "Mexico City" "Munich"

[21] "Montreal" "Moscow" "Los Angeles" "Seoul" "Barcelona"

[26] "Atlanta" "Sydney" "Athens" "Beijing" "London"

[31] "Rio de Janeiro" "Tokyo"

**删除所有小括号和中括号**

*# remove text inside parenthesis*

cities <- gsub(" \\(.\*?\\)", "", countries)

*# remove text inside square brackets*

cities\_cleaned <- gsub(" \\[.\*?\\]", "", cities)

**改多列名**

colnames(ed) <- c('GM',1993:2017)

改一列名

names(df)[names(df) == 'old.var.name'] <- 'new.var.name'

**把某一列按照’：’separate按照某一指标分开**

ed2 <- ed1 %>% separate(GM, c('Gender','Major'), sep = ": ", remove = TRUE, convert = TRUE)

**把dataframe中所有的空格都删掉**

ed2$Gender <- trimws(ed2$Gender)*#把前后的空格删掉*

**把dataframe中有，的数字变成数字格式**

ed2$Count <- gsub(',','',ed2$Count)

ed2$Count <- trimws(ed2$Count)

a[is.na(a)] <- 0 *#把na都变成0*

ed2$Count <- as.numeric(ed2$Count)

**最大的，返回名称**

d3 <- d2 %>% group\_by(Year) %>% summarise(number=sum(Count))

**求vector中每个element的长度用nchar**

nchar(d6$year)==2])

**再ggplot上面搞个标签**

library(ggrepel)

ggplot(df2,aes(x=mean\_pledge,y=success\_rate,color=main\_category,size=median\_goal))+

geom\_point()+

**geom\_label\_repel**(aes(label = main\_category), size=2.5,box.padding=0.3)+

xlab('Average Amount Pledged per Backer (USD) ') +

ylab('Success Rate')

# lesson7

**关于full loop**

for(i in 1:10)

{

word <- paste('Word:',i)

print(word)

}

**function和simulation**

PlayOneGame <- function(iter)

{

doors <- c('Door1','Door2','Door3')

#step1: Host decide what door to put the prize

prize.door <- sample(doors,1)

#step2: Player choose the door

chosen.door <- sample(doors,1)

#step3: Host open a door

remain.doors <- doors[doors != prize.door & doors != chosen.door]

open.door <- sample(remain.doors,1)

#step4: swap to another door

swap.door <- doors[doors != open.door & doors != chosen.door]

#Return T if swap door wins the price, F otherwise

return(prize.door == swap.door)

}

n <- 10000

totalWins <- 0

for (i in 1:n)

{

totalWins <- totalWins + PlayOneGame(i)

}

prob.winning.by.swapping <- totalWins / n

prob.winning.by.swapping

**对xxxx重复使用同样的公式，生成的是一个vector**

sapply(1:n,PlayOneGame)

# lesson8

**画smooth图**

library(ggplot2)

ggplot(data=f,aes(x=Foot, y=Height)) + geom\_point() + geom\_smooth(method='lm') + theme\_bw()

**linear regression model**

model <- lm(Height ~ Foot, data=f)

summary(model)

**预测 predict（），用算出来的model来推测新的值（注意预测的data要有因变量的那一列）**

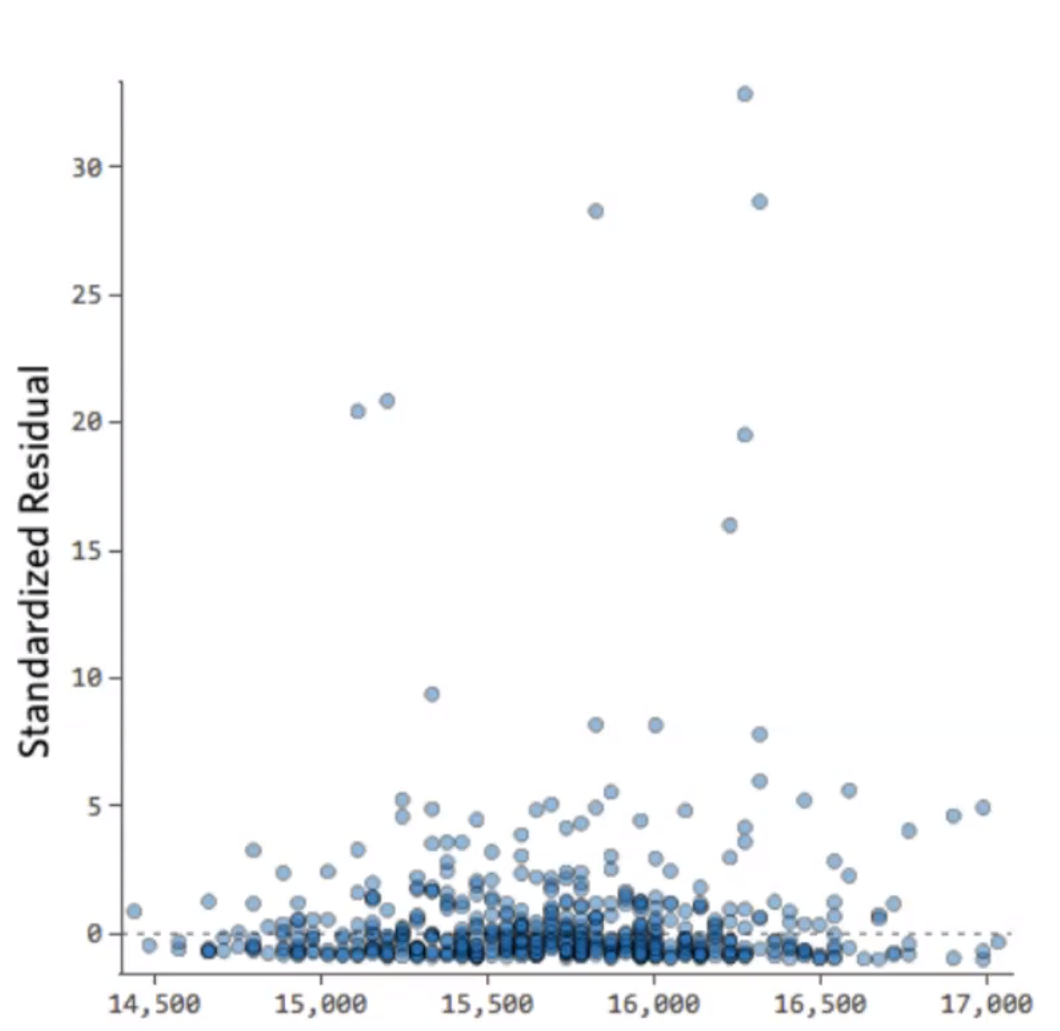
f$Predicted.Height <- predict(model, f)

head(f)

**可能会遇到的情况 standardized residual标准化残差**

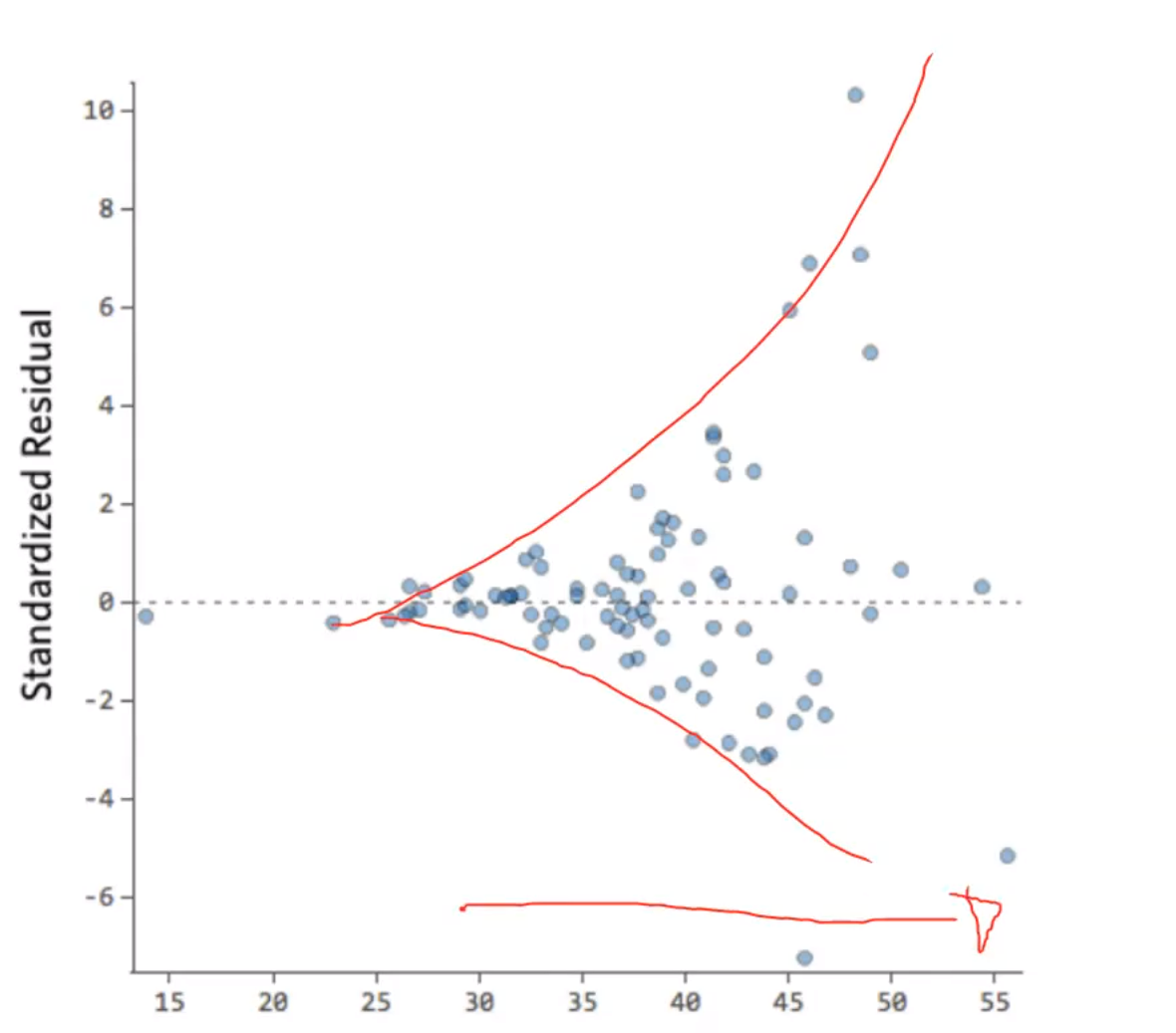
**1.有outliers，有些变量的residual不是normal distribution**

**要用log去transform the data**

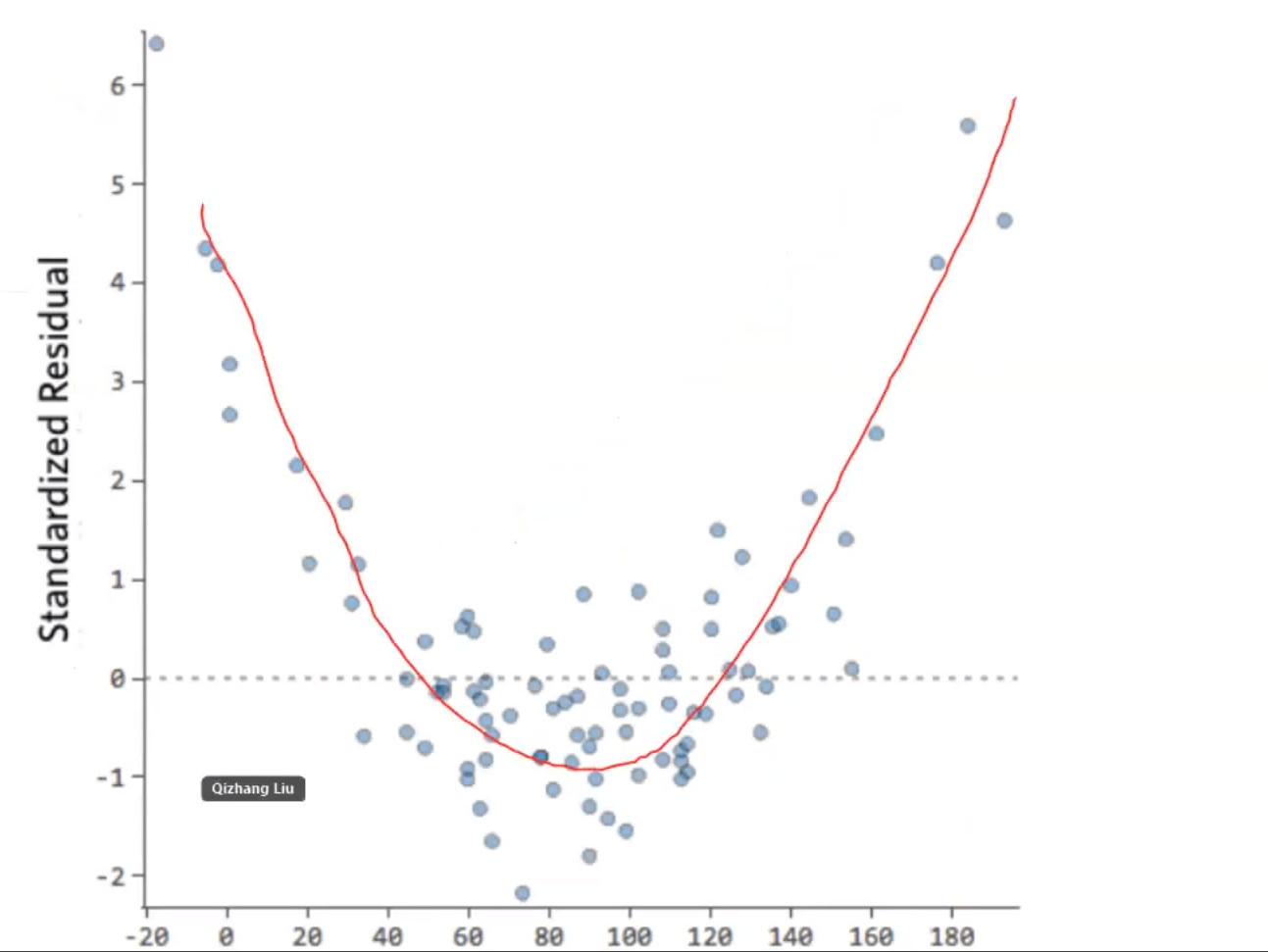


**2. 没有homoscedasticity(equal variance)同方差性**

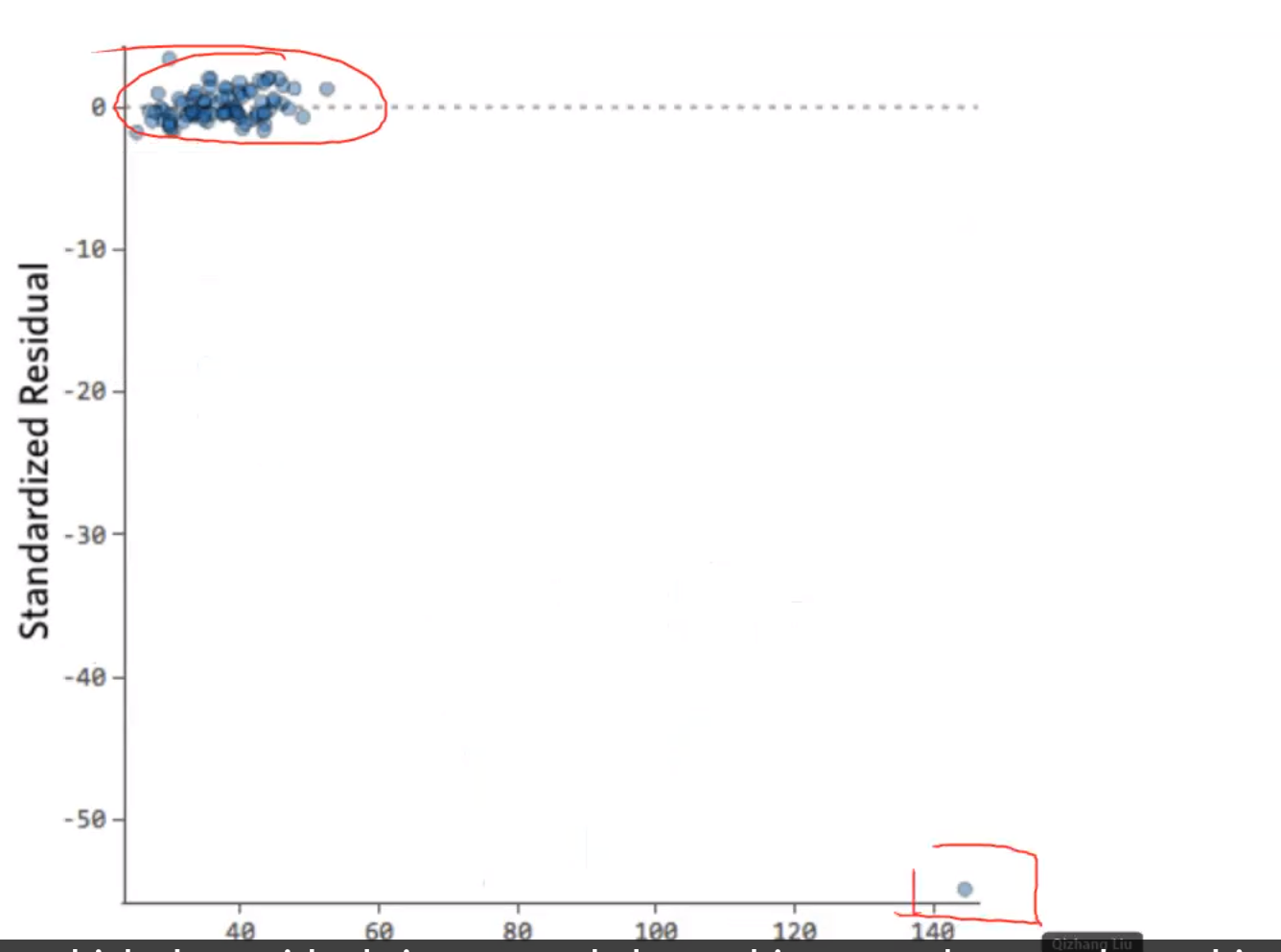
**当x很小的时候error小，而当x变大的时候，error就越来越大**



3.nolinear



4.outliers



**加平方项**

model2 <- lm(Height ~ Foot + **I(Foot^2)**,data=f) #加了square turn

summary(model2)

**多元线性回归**

model <- lm(overall**~.**, data=data) 有个点

summary(model)

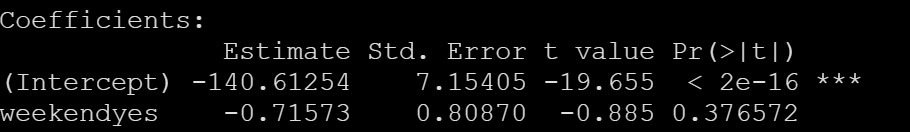
**去掉某个variable**

model2 <- lm(overall~.**-weekend**, data=data)

summary(model2)

**dummy variable**

这里weekendyes的Estimate的意义是，when all the other variable remain the same, those visiting during weekend will have 0.71573 point lower feedback than those in weekdays. 永远是dummy variable和base value做对比。



**Interaction effect**相互影响，交互项

model3 <- lm(overall~.**+weekend\*distance**,data=data)

summary(model3)

# Lesson10

**随机选random**sample(doors,1)

sample <- sample(c(TRUE,FALSE),nrow(triers),prob = c(0.75,0.25),replace = TRUE)

triers.train <- triers[sample, ]

triers.test <- triers[!sample, ]

**logistic regression逻辑回归的图**

**1.关于Continous variable**

ggplot(triers)

geom\_point(aes(x=Age,y=HaveTried),position=position\_jitter(width=0.1,height = 0.1)) + geom\_smooth(aes(x=Age,y=as.numeric(HaveTried)+1))

设置function来求最优的threshold

PredictErrorByCutContinousVar <- function(threshold, data, predictor, outcome){

predict <- factor(data[,predictor] < threshold, levels=c("FALSE","TRUE"))

table <- table(predict = predict, actual = data[,outcome])

table

return ((table[1,2]+table[2,1])/sum(table))

}

range(triers$Age)

ages<-23:64

errors <- sapply(ages,PredictErrorByCutContinousVar,triers,"Age","HaveTried")

df <- data.frame(Age=ages,Error=errors)

optimal.Age <- df[which.min(df$Error),"Age"]

errors <- sapply(ages,PredictErrorByCutContinousVar,triers,"Age","HaveTried")

df <- data.frame(Age=ages,Error=errors)

ggplot(df, aes(x=Age,y=Error)) + geom\_line() + geom\_vline(xintercept = optimal.Age, linetype="dotted",color = "red") + geom\_text(aes(x=(optimal.Age+1), label=paste0("Optimal Cut-off Age: ",optimal.Age), y=0.45), colour="red", angle=90, text=element\_text(size=11))

**设置阈值threshold来分析**

predict <- triers[,"Age"] < 35

predict <- factor(predict,levels=c("FALSE","TRUE"))

table <- table(predict = predict, actual = triers[,"HaveTried"]) 前面是竖着，后面是横着

table

#Error rate

(table[1,2]+table[2,1])/sum(table)

#accuracy

(table[1,1]+table[2,2])/sum(table)

2.关于Categorical Variable

ggplot(triers,aes(x=PayType,fill=HaveTried)) + geom\_bar(position="fill", width = 0.5)

table <- table(triers$PayType == "Salaried", triers$HaveTried)

error <- (table[1,2]+table[2,1])/sum(table)

取对数意味着原被解释变量对解释变量的弹性，即百分比的变化而不是数值的变化；  
目前，对于什么时候取对数还没有固定的规则，但是有一些经验法则：  
（1）与**市场价值**相关的，例如，**价格、销售额、工资**等都可以取对数；  
（2）以年度量的变量，如受**教育年限、工作经历**等通常**不取对数**；  
（3）比例变量，如失业率、参与率等，两者均可；  
（4）变量取值必须是非负数，如果包含0，则可以对y取对数ln(1+y);  
取对数的一个缺陷就是难以预测原变量的值。因为对数模型中预测的是lny,而不是y.

**画图的例子汇总**

**1.求两个变量之间的关系：先求density，再画图 点图**

cs$deadline <- as.Date(cs$deadline)

cs$launched <- as.Date(cs$launched)

cs$duration <- as.numeric(cs$deadline - cs$launched)

cs <- cs[cs$launched > as.Date('2000-01-01'),]

#求这两个因素的关系，先求density，看看哪个log10之后能normal

ggplot(cs,aes(x=duration)) + geom\_density()

ggplot(cs,aes(x=usd\_pledged\_real)) + geom\_density()

ggplot(cs,aes(x=log10(usd\_pledged\_real))) + geom\_density()

#所以要scale\_y\_log10()

ggplot(data=cs,aes(x=duration, y=usd\_pledged\_real)) + geom\_point(alpha=0.1) + geom\_smooth(color='blue') + scale\_y\_log10()

**2.求不同类别的，3个变量之间的关系：一般都是color=category，size是第3个变量 点图**

分析的时候说：y比较大的类别，y大的原因；y比较小的类别，y小的原因。

Analysing the two charts, it seems that Dance and Theater has higher rate of success due to larger average amount pledged per donor as well as large number of backers, as well as generally a smaller, more achievable goal. Comics has a high success rate mainly due to large number of backers and smaller goal. Technology usually has a large goal and relatively small number of backers, thus it is more likely to fail its fundraising target.

library(ggrepel)

df2 <- cs %>% group\_by(main\_category) %>% summarise(success\_rate=sum(state=='successful')/n(),

mean\_pledge=mean(ppb),

median\_goal=median(usd\_goal\_real))

ggplot(df2,aes(x=mean\_pledge,y=success\_rate,color=main\_category,size=median\_goal))+

geom\_point()+

geom\_label\_repel(aes(label = main\_category), size=2.5,box.padding=0.3)+

xlab('Average Amount Pledged per Backer (USD) ') +

ylab('Success Rate')

**3.在2上面扩展，又增加了2个变量。相当于探索4个x和y之间的关系。这时候画两个图。 点图**

cs$apg <- cs$ppb/cs$usd\_goal\_real

df2 <- cs %>% group\_by(main\_category) %>% summarise(median\_backers=median(backers),median\_goal\_contribution=median(apg))

ggplot(df2,aes(x=median\_goal\_contribution,y=success\_rate,color=main\_category,size=median\_backers))+

geom\_point()+

geom\_label\_repel(aes(label = main\_category), size=2.5,box.padding=0.3)+

xlab('Average Percentage of Goal Contributed per Backer') +

ylab('Success Rate')

**4.求按照时间的变化 横坐标是时间 折线图**

plot a line chart to show how the overall average monthly amount pledged varied

df1$launched <- as.Date(paste0(**format(**as.Date(df1$launched), "%Y-%m"**)**,'-01'))

ggplot(data=df1) + geom\_line(aes(x=launched,y=usd\_pledged\_real),stat='summary',fun.y='median') +

xlab('Launched date') +

ylab('Monthly median Amount Pledged')

**5.比较不同种类，某个指标的平均值 柱状图**

cs$ppb <- ifelse(cs$backers>0,cs$usd\_pledged\_real/cs$backers,0)

#总的average

a <- mean(cs$ppb)

#在bar的形式下，先将y变量求和，再用mean显示

#再加一条平均线

ggplot(data=cs,aes(x=reorder(main\_category,ppb),y=ppb)) + geom\_bar(stat = 'summary', fun.y = 'mean') + theme(axis.text.x=element\_text(angle=90)) + geom\_hline(yintercept=a,colour = 'blue', size = 0.7) + labs(y='Average amount pledged per backer', x='Category')

**6.普通的比较不同种类 某个变量的区别 点图 加boxplot**

ggplot(data=cs,aes(x=main\_category,y=usd\_goal\_real)) + geom\_point() + geom\_boxplot(color='black') + coord\_cartesian(ylim = c(0,60000)) + theme(axis.text.x=element\_text(angle=90))

7.展现两个指标的某个数量各有多少 **柱状图**

df <- p %>% group\_by(SUBJECT\_RACE, TIME.SCENARIO) %>% summarise(number\_of\_cases=n(),.groups = 'drop')

ggplot(data=df,aes(x=SUBJECT\_RACE,y=number\_of\_cases,fill=TIME.SCENARIO)) + geom\_bar(stat='identity',position='dodge')

country\_coord[(rownames(country\_coord)==c('Portugal')) | (rownames(country\_coord)==c('United Kingdom')) | (rownames(country\_coord)==c('France')) | (rownames(country\_coord)==c('Spain')) | (rownames(country\_coord)==c('Germany'))]