**Week8**

###Mining Time Series Data

set.seed(123)

data1<-rnorm(12)

#create time series data as quarterly data starting

#from 2nd quarter of 2022也就是按季度排，从22年第二季开始

X<-ts(data1, frequency = 4,start=c(2022,2))

#create monthly time series data starting

#from February 2021也就是按月份排，21年2月开始

X2<-ts(data1,frequency = 12, start=c(2021,2))

#检查变量类别用class

class(data1)

class(X)

X2

#r adopting ISO 8601 format

#系统的时间，也就是现在电脑显示的日期和时间

date<-Sys.Date()

time<-Sys.time()

date2<-as.Date("2023-5-12")

date2

class(date2)

#create date starting from 2016-01-01 on to 2018-12-31

daily\_index<-seq.Date(from=as.Date("2016-01-01"),

to=as.Date("2018-12-31"),by="day")

#3-day interval三天一个数据

day3\_index<-seq.Date(from=as.Date("2016-01-01"),

to=as.Date("2018-12-31"), by="3 day")

#data dates.formats2.csv

data\_df<-read.csv(file.choose(),stringsAsFactors = F,

sep=";")

head(data\_df)

class(data\_df)

str(data\_df)

#transform character to date format转化日期格式

attach(data\_df)

US\_format\_New<-as.Date(US\_format,format="%m/%d/%Y")

US\_format\_New

Japanese\_format\_new<-as.Date(Japanese\_format,format="%d/%m/%Y")

Japanese\_format\_new

#Excel\_Numeric\_Format

#in excel1, the origin of date is December 30, 1899

#in R, the origin January 1, 1970

#Excel数字日期格式转换为 R 中的日期格式,origin指定了Excel中日期的起点

Excel\_f<-as.Date(Excel\_Numeric\_Format,

origin=as.Date("1899-12-30"))

Excel\_f

Excel\_Numeric\_Format

#Class time series data in R

#POSIX class

time\_str<-"2017-01-20 23:59:34"

time\_str\_ctl<-as.list.POSIXct(time\_str)

#zoo

install.packages("TSstudio")

library(TSstudio)

library(zoo)

data(US\_indicators)

str(US\_indicators)

Vehicle\_Sales1<-zoo(x=US\_indicators$'Vehicle Sales',

frequency=12)

Vehicle\_Sales1

#ts class(下面那坨是TSstudio包的，可以敲这个ts\_plot(USgas)查看)

data(USgas)

class(USgas)

#xts class

library(xts)

version#检查版本用的，有些包可能要更新才能用

Vehicle\_Sales2<-xts(x=US\_indicators$'Vehicle Sales',

frequency=12,order.by=US\_indicators$Date)

Vehicle\_Sales2

#####Time series decomposition#####

#seasonal component

data(USgas)

ts.plot(USgas, main="US Monthly Natural Gas consumption",

ylab="Billion Cubic Feet",xlab="year")#时间序列线图

ts\_heatmap(USgas)#时间序列热图

#cycle component

data(USUnRate)

ts.plot(USUnRate, main="US Monthly Unemployment Rate",

ylab="Unemployment Rate",xlab="year")

ts\_heatmap(USUnRate)

#Types of decomposition

#additive structure

ts.plot(USgas, main="US Monthly Natural Gas consumption",

ylab="Billion Cubic Feet",xlab="year")

usg\_decompose<-decompose(USgas)#时间序列分解，加法结构

plot(usg\_decompose)

#extract each component 列出各个组件的名称

names(usg\_decompose)

#seasonal component

usg\_decompose$seasonal

#trend component

usg\_decompose$trend

#random component

usg\_decompose$random

#multiplicative structure

data(AirPassengers)

ts.plot(AirPassengers, main="Monthly Airline Passenger",

ylab="Thousand of Passengers",xlab="year")

air\_decompose<-decompose(AirPassengers,

type="multiplicative")#分解，乘法结构

plot(air\_decompose)

#extract each component

names(air\_decompose)

#seasonal component

air\_decompose$seasonal

#trend component

air\_decompose$trend

#random component

air\_decompose$random

#convert multiplicative structure to additive structures.乘法变加法

library(forecast)

#multiplicative structure

data(AirPassengers)#名为 AirPassengers 的时间序列数据

#计算 AirPassengers 数据的 Box-Cox 变换的 lambda 值

AirP<-BoxCox.lambda(AirPassengers)

#根据之前计算的 lambda 值变换。旨在将数据转换为对数形式

AirPT<-BoxCox(AirPassengers,lambda = AirP)

par(mfrow=c(1,2))#一行两列，以便同时显示两张图

plot.ts(AirPassengers)

plot.ts(AirPT)

###Time series forecasting

#data Tovel.csv

data<-read.csv(file.choose(),header=T)

yt<-ts(data)

#图片边界par(mfrow=c(2,1))，这个可能大了，我换了个显示尺寸

par(mar = c(1, 1, 1, 1))

plot.ts(yt, main="Paper Towel Daily Sales",ylab="yt")

#data is not stationary不平稳

#do a differentation差分操作用于使数据更加平稳

#diff() 函数进行差分操作，减去前一个时间点的值来消除趋势或周期性，使数据更平稳

ztl<-diff(yt,differences=1)

plot.ts(ztl, main="1st difference data",ylab="zt")

#ARIMA model identification

#自相关函数（ACF）和偏自相关函数（PACF），用于帮助识别 ARIMA 模型的参数。

#自相关函数显示了序列与其自身滞后版本之间的相关性，帮助识别时间序列中可能的自相关结构。

acf(ztl, main="Sample Autocorrelation Function")

#偏自相关函数显示了序列在移除其它滞后效应后与当前滞后的相关性，有助于确定时间序列中的部分自相关结构。

pacf(ztl,main="Sample Partial Autocorrelation Function")

#based on ACF & PACF plot

#the model should be ARIMA(0,1,1)

#fit ARIMA(0,1,1) to dataset

#arima() 函数拟合了 ARIMA(0,1,1) 模型到时间序列数据 yt 上

#order = c(0, 1, 1) 指定了模型的阶数，其中 0 表示非季节性自回归（AR）阶数

#"1" 表示非季节性移动平均（MA）阶数

model<-arima(yt,order=c(0,1,1))

summary(model)

#plot model to observed dataset

plot.ts(yt,main="Paper Towel Daily Sales",ylab="yt")

lines(fitted(model),col="red",lty=2)#拟合值

legend("bottomleft",

c("Observed Data","Model ARIMA(0,1,1)"),

col=c(1,2),lty=c(1,2))#图例

#Diagnostic check

#residual analysis

library(forecast)

f.value<-forecast(model,h=5)

residual<-f.value$residuals

#residual should not be autocorelated模型的残差（residual）的自相关函数图

acf(residual, lag.max = 20)

#based of ACF plot

#residual is not autocorelated

#residual should approximate normal distribution

#通过观察残差的直方图，可以初步检查残差是否近似于正态分布

hist(residual)

#the residual should having a constant variance over time series

#残差随时间变化的图形。这有助于观察残差是否在时间序列上保持恒定的方差

plot.ts(residual)

#since all the assumptions is fulfiled, the ARIMA(0,1,1)

#model can be used for prediction

#残差看起来在自相关性、正态性和方差稳定性方面都表现得很好，

#这意味着你的 ARIMA(0,1,1) 模型对于预测是合适的。

#这种检查有助于确认模型是否能够准确地捕捉数据的特征，

#并且验证了模型是否可以用于进一步的预测。

#Forecasting from ARIMA(0,1,1)model

plot.ts(yt, main="Paper Towel Daily Sales",ylab="yt")

lines(fitted(model),col="red",lty=2)

forecasting<-predict(model,n.ahead = 5)# 5 步的预测

#50% confidence interval常见的 50% 置信区间

U<-forecasting$pred+0.69\*forecasting$se

L<-forecasting$pred-0.69\*forecasting$se

#forecasting plot图形中加入了图例，标注了观测数据、预测值和置信区间

ts.plot(yt,forecasting$pred,U,L,col=c(1,2,4,4),lty=c(1,1,2,2))

legend("bottomleft",c("Observed Data","Forecast","Confidence Interval (50%)"),

col=c(1,2,4),lty=c(1,1,2))

#Time series clustering时间序列聚类

#打开week8中的sample2谢谢喵

#load("D:/RSTUDIO/R project/Week8/sample2.RData")储存位置可改，或File直接开

ts.plot(sample2)

par(mfrow=c(3,3))#绘图区域分割为 3 行 3 列

#plot.ts() 函数来逐一绘制 sample2 中的不同时间序列变量，每个变量占一个小图

plot.ts(sample2[,1]);plot.ts(sample2[,2]);plot.ts(sample2[,3])

plot.ts(sample2[,4]);plot.ts(sample2[,5]);plot.ts(sample2[,6])

plot.ts(sample2[,7]);plot.ts(sample2[,8]);plot.ts(sample2[,9])

#没有就下包喵~#determine the cluster动态时间规整

install.packages("dtw")

library(dtw)

#put a label on each time series data

#创建了标签，用于表示每个时间序列数据点所属的类别

D.Labels<-rep(1:60)

#compute distance matrix

#使用 dist() 函数计算了时间序列数据集 sample2 之间的距离矩阵。

#method = "DTW" 表示使用动态时间规整（DTW）方法计算距离

distMatrix<-dist(sample2,method="DTW")

#plot the cluster

#使用 hclust() 函数进行层次聚类，根据时间序列之间的距离构建聚类结构。

#method = "average" 表示使用平均链接法

TSCluster<-hclust(distMatrix,method = "average")

plot(TSCluster,labels=D.Labels,

main="Time Series Clustering")

#get 6 cluster将数据分为 6 类的矩形框

rect.hclust(TSCluster,k=6)

#justify your results

#cluster聚类

window(10,10)

par(mfrow=c(2,4))

plot.ts(sample2[,34]);plot.ts(sample2[,35]);plot.ts(sample2[,32])

plot.ts(sample2[,37]);plot.ts(sample2[,33]);plot.ts(sample2[,31])

plot.ts(sample2[,36])

window(10,10)

par(mfrow=c(1,3))

plot.ts(sample2[,18]);plot.ts(sample2[,16])

plot.ts(sample2[,20])

#Time series classification

install.packages("party")

library(party)

#创建了一个类别标签 classId，共有 6 个类别，每个类别包含 10 个样本

classId<-rep(as.character(1:6),each=10)

#打开Data.Ks喵谢谢喵#ctree() 函数进行分类建模，评估了分类模型的预测效果。

Class.Model<-ctree(classId~.,data=Data.Ks)

plot(Class.Model)#绘制了构建的条件推断树模型的可视化结果

#measure precison of prediction model

pClassId<-predict(Class.Model)#得到预测类别

#计算了实际类别和预测类别之间的交叉表，可以帮助评估模型的预测效果

table(classId,pClassId)

#prediction accuracy

#计算模型的预测准确率，即预测正确的样本数量与总样本数量之比

sum(classId==pClassId)/nrow(Data.Ks)

#这些步骤帮助你构建了一个分类模型，并且评估了该模型对于给定数据的分类准确度。

#通过这些指标，你可以初步了解模型的预测效果和分类准确度。

**###########################################**

**Week10**

##mining sequence data

install.packages('TraMineR')

library(TraMineR)

#example of sequence data

data(mvad)

summary(mvad)#摘要

##define a sequence data

#define a labels and codes of codes for state

#例如:"employment"被定义为"EM"

mvad.labels<-c("employment","further education",

"higher education","joblessness",

"school","training")

mvad.scode<-c("EM","FE","HE","JL","SC","TR")

#create sequence object in R

#使用了从第17列到第86列的数据来创建序列对象，并使用之前定义的状态代码和标签。

mvad.seq<-seqdef(mvad, 17:86, states=mvad.scode,

labels=mvad.labels)

#statistical summary indicators

#mean time spent in each state

#overall dataset每种状态（就业、进修、失业等）的平均持续时间

seqmeant(mvad.seq)

#mean time spent in each state by group

#group:fathers employment status

#by()函数，这次是按照"XX"对数据进行分组，

#并计算了每个分组内每种状态的平均持续时间。

#这可以用来比较不同XX情况的年轻人在每种状态上的平均持续时间。

by(mvad.seq, mvad$funemp, seqmeant)

#group:gender

by(mvad.seq, mvad$male, seqmeant)

#number of transition状态转换的次数

nT<-seqtransn(mvad.seq)

hist(nT)#直方图

#transition rates转换率

#overall dataset

mvad.trate<-seqtrate(mvad.seq)

#if the transition rate different over time

#time varying transition rates

mvad.trate2<-seqtrate(mvad.seq, time.varying = T)

##Visualizing sequence data

#sequence index plot序列索引图

#每一条线代表一个序列，横轴表示序列的时间顺序，

#纵轴表示数据集中的序列编号。这有助于观察序列在数据集中的分布情况

seqiplot(mvad.seq, border=NA,

main="Sequence index plot")

#sequence frequency plot序列频率图

seqfplot(mvad.seq, border=NA,

main="Sequence frequency plot")

#sequence distribution plot序列分布图

#观察数据集中序列的多样性和分布情况

seqdplot(mvad.seq, border=NA,

main="Sequence distribution plot")

#model state plot模态状态图

#展示了每个时间点上最常见的状态

seqmsplot(mvad.seq, border=NA,

main="Modal state plot")

#transversal entropy横向熵图

#横向熵是一种度量序列模式多样性的指标，

#它表示在给定时间点上不同序列的多样性程度

seqHtplot(mvad.seq, main="Transversal Entropies")

#event sequence of transition

#define event:sequence of transition

mvad.seqe<-seqecreate(mvad.seq)

#look for frequent event subsequences

#seqefsub() 函数通过事件序列 mvad.seqe 找到频繁出现的事件子序列，

#参数 pmin.support = 0.05 表示最小支持度为 0.05，即出现频率至少为 5%

fsubseq<-seqefsub(mvad.seqe, pmin.support = 0.05)

#plot 15 most frequent event前15个最频繁的事件子序列

plot(fsubseq[1:15], col="blue")

##Categorizing pattern in sequence data聚类算法

library(cluster)

#compute optimal matching distance based on transition rate

#这行代码计算了基于转换率的最优匹配距离。

#seqsubm() 函数根据指定的方法（这里是"TRATE"，即基于转换率），

#计算了序列对象 mvad.seq 中每对序列之间的相似性矩阵。

#这个相似性矩阵描述了序列之间的相似程度

submat<-seqsubm(mvad.seq, method="TRATE")

#这行代码使用最优匹配距离（Optimal Matching，OM）来计算序列数据的距离矩阵。

#seqdist() 函数计算了基于最优匹配的序列之间的距离，

#并使用了之前计算的相似性矩阵 submat。这个距离矩阵描述了序列之间的差异程度。

dist.om<-seqdist(mvad.seq, method="OM", sm=submat)

#build hierarchical clustering based on

#optimal matching distance

#agnes()函数进行聚类分析。

#agnes()函数执行了AGNES（AGglomerative NESting）层次聚类算法，

#它接受一个距离矩阵 dist.om 作为输入，并使用ward方法来构建聚类。

#diss = TRUE 表示输入的是距离矩阵

clusterward<-agnes(dist.om, diss=T, method="ward")

plot(clusterward)#按<Return>键来看下一个图

#retrieve each sequence data for some cluster

#example:k=4 cluster

#cutree()将层次聚类的结果切割为指定数量的聚类， k = 4，即切割成4个聚类。

cl.4<-cutree(clusterward, k=4)

#每个聚类分配了一个标签

#用了paste()函数来生成标签，例如："type 1"、"type 2"、"type 3"、"type 4"

cl.4fac<-factor(cl.4, labels=paste("type",1:4))

seqiplot(mvad.seq,group=cl.4fac, border=NA,

main="Sequence index plot")

seqfplot(mvad.seq,group=cl.4fac, border=NA,

main="Sequence frequency plot")

seqdplot(mvad.seq,group=cl.4fac, border=NA,

main="Sequence distribution plot")

seqmsplot(mvad.seq, border=NA,group=cl.4fac,

main="Modal state plot")

seqHtplot(mvad.seq,group=cl.4fac, main="Transversal Entropies")

#Determine the most discriminating transitions

#between cluster and plot the frequencies by cluster

#seqecmpgroup()函数来确定不同聚类之间具有区分性的转换。

#它分析了之前找到的频繁事件子序列 fsubseq 在每个聚类中的出现频率差异，

#从而确定哪些转换最能区分不同的聚类。group = cl.4fac 指定了要分析的聚类变量。

discr<-seqecmpgroup(fsubseq, group=cl.4fac)

plot(discr[1:6])#频率最高的前6个

**###########################################**

**Week11**

#Mining Text Data#

install.packages("tm")

install.packages("SnowballC")

install.packages("wordcloud")

install.packages("RColorBrewer")

install.packages("syuzhet")

install.packages("ggplot2")

library(tm)#文本挖掘

library(SnowballC)

library(wordcloud)#词云

library(RColorBrewer)

library(syuzhet)

library(ggplot2)

#import the data选择text文件

text<-readLines(file.choose())

class(text)

#define a corpus text创建了一个文本语料库（corpus）

docs<-Corpus(VectorSource(text))

class(docs)

inspect(docs)#查看创建的语料库的内容

#Data cleaning of text data

#(i)remove special character from text

#define blank space

#替换为空格

toSpace<-content\_transformer(function(x,pattern)

gsub(pattern,"",x))

#convert all special character into blank space

docs2<-tm\_map(docs,toSpace,"/")

docs2<-tm\_map(docs2,toSpace,"@")

docs2<-tm\_map(docs2,toSpace,"!")

docs2<-tm\_map(docs2,toSpace,",")

docs2<-tm\_map(docs2,toSpace,":")

#(ii)Convert the text to lower case小写

docs2<-tm\_map(docs2,content\_transformer(tolower))

#(iii)remove numbers移除文本中的数字

docs2<-tm\_map(docs2,removeNumbers)

#(iv)remove the stopwords移除英文停用词

docs2<-tm\_map(docs2,removeWords,

stopwords("english"))

#(v)remove punctuation移除标点符号

docs2<-tm\_map(docs2,removePunctuation)

#(vi)eliminate extra unnecessary spaces in the text

#消除文本中额外的空格

docs2<-tm\_map(docs2,stripWhitespace)

###Word tokenization

library(tidyverse)

library(tokenizers)

#tokenize\_words函数对文本进行了词语分词化，

#将文本拆分成单词列表，保存在words.Tk变量中

words.Tk<-tokenize\_words(text)

#Text stemming提取词干（stem），即将单词转换为其词根形式，存储在docs3变量

docs3<-tm\_map(docs2,stemDocument)

inspect(docs3)#展示了处理后的文本数据

#Document-term matrix

#TermDocumentMatrix函数将经过处理的文本数据集docs3转换为文档-词项矩阵（DTM），

#将单词的出现频率映射到一个矩阵中，保存在dtm变量中

dtm<-TermDocumentMatrix(docs3)

m<-as.matrix(dtm)

m

#the frequency table of words词频表

#rowSums(m)用于计算每个单词在整个文本数据集中的总出现次数

#sort(..., decreasing = TRUE)对单词按照频率进行排序

v<-sort(rowSums(m),decreasing = T)

#word列存储单词，freq列存储对应的频率

d<-data.frame(word=names(v),freq=v)

#plot word frequency

#first 20 most frequent words最高的前20个单词的频率

#las=2参数将标签文本旋转90度

#names.arg=d[1:20,]$word设置了条形图中每个条形的标签，前20个高频单词内容

barplot(d[1:20,]$freq,las=2, names.arg=d[1:20,]$word,

main="Most frequent words")#main="Most frequent words"图表标题

#word cloud

set.seed(12)

#words=d$word是用于词云的单词列表，freq=d$freq包含了每个单词的频率信息

#min.freq=2设置词云中单词最小出现频率，max.words=100定义了词云中最大单词量

#random.color=F禁用了随机颜色

wordcloud(words=d$word, freq=d$freq, min.freq=2,

max.words = 100,random.color=F,

color=brewer.pal(8,"Dark2"))

#Word association（findAssocs函数用于分析词语之间的关联性）

#analyze the association between frequent words

#Example:which words are associated "freedom"?

#freedom”相关联的单词，其相关性阈值为0.3

findAssocs(dtm,terms="freedom",corlimit=0.3)$freedom

#Example:association between some particular words

#展示三个单词之间的相关性

findAssocs(dtm,terms=c("freedom","dream","will"),corlimit=0.25)

#Example:find association for words that occur

#at least 10 times至少出现10次的词

findAssocs(dtm,terms=findFreqTerms(dtm,lowfreq = 10),corlimit = 0.25)

##Sentiment Analysis

library(sentimentr)

x<-'Sentiment analysis is super fun'

sentiment(x)#正面情感

y<-'Sentiment analysis is super boring. I do love work'

sentiment(y)#包含正面和负面情感的复合情感

#get\_sentiment函数则用于对整个文本进行情感分析

#method="syuzhet"来采用了"syuzhet"方法进行情感分析

sentiment\_sz<-get\_sentiment(text,method="syuzhet")

hist(sentiment\_sz)

summary(sentiment\_sz)

##Emotion classification用了NRC情感词典（NRC Sentiment Lexicon）

#get\_nrc\_sentiment函数用于基于NRC情感词典对文本进行情感分类

#列代表不同的情感类别，行代表文本中的每个词或短语

Ec<-get\_nrc\_sentiment(text)

td<-data.frame(t(Ec))#计数

td\_new<-data.frame(rowSums(td))#新数据框，每行表示一种情感类别的总计数

td\_new

##transformation & cleaning

names(td\_new)[1]<-"count"#数据框td\_new的第一列更改列名为"count"

#将数据框td\_new的行名（情感类别）添加为一列，并命名为"sentiment"

td\_new<-cbind("sentiment"=rownames(td\_new),td\_new)

rownames(td\_new)<-NULL#td\_new的行名设为NULL，将其重置

td\_new2<-td\_new[1:8,]#只显示前8个

#条形图的高度代表了每种情感类别的计数，通过不同的颜色进行区分

quickplot(sentiment, data=td\_new2,weight=count,

geom="bar",fill=sentiment,ylab="count")+

ggtitle("Survey Sentiments")

#Mining Spatial Data

library(sp)

library(raster)

load("D:\\RSTUDIO\\R project\\week11\\wst.RData")

attach(wst)#变量直接放入当前环境

lonlat<-cbind(longitude,latitude)#数据集的经度和纬度组成一个矩阵lonlat

pts<-SpatialPoints(lonlat)#经纬度坐标点

class(pts)

plot(pts)

#define coordinate system

crdref<-CRS('+proj=longlat +datum=WSG84')#定义了坐标系

#坐标系信息应用到 SpatialPoints 对象

pts<-SpatialPoints(lonlat,proj4string = crdref)

#embed the atribut to SpatialPoints vector

df<-data.frame(ID=name,precip)#新数据框包含了名称和降水量

ptsdf<-SpatialPointsDataFrame(pts,data=df)

plot(ptsdf)

showDefault(ptsdf)

#spatial lines data这里spLines()函数用于创建空间线数据

lonlat<-cbind(longitude,latitude)

linesD<-spLines(lonlat,crs=crdref)

plot(linesD)

#spatial polygom data创建空间多边形数据

lonlat<-cbind(longitude,latitude)

polyD<-spPolygons(lonlat,crs=crdref)

plot(polyD)

#raster form data创建栅格形式的空间数据

r<-raster(ncol=10,nrow=10,xmx=-80,xmn=-150,

ymn=20,ymx=60)

area<-rnorm(ncell(r))#随机数，范围r

values(r)<-area

plot(r)

#raster stack & raster brick

#stack() 函数将这四个栅格对象组合成一个栅格堆叠（stack）。

#栅格堆叠是将多个栅格对象堆叠到一个对象中，每个栅格对象对应于堆叠中的一个层

r2<-r\*r

r3<-r^3

r4<-2\*r2+r3

s<-stack(r,r2,r3,r4)

plot(s)

#data manipulation in spatial analysis

#下面找到了 raster 包中的 lux.shp 文件的路径，矢量空间数据文件 lux.shp

f<-system.file("external/lux.shp",package = "raster")

p<-shapefile(f)

#presenting data in data frame format

d<-data.frame(p)

#extract spesific atributes提取空间数据对象 p 中的特定属性

p$NAME\_1

p$AREA

#add new variable

set.seed(12)

xp<-sample(letters,length(p))

p$new<-xp#生成的随机字母序列 xp 添加为 p 对象的一个新变量 new

data.frame(p)#创建了一个数据框，显示了经过变量添加和删除后的空间数据对象 p

#delete any variable删除了 p 对象中的 new 变量

p$new<-NULL

#data integration

dfr<-data.frame(District=p$NAME\_1,Canton=p$NAME\_2,

Value=runif(length(p)))

cD<-merge(p,dfr,by.x=c('NAME\_1','NAME\_2'),

by.y=c('District','Canton'))

#data aggregation绘制地图

pa<-aggregate(p,by='NAME\_2')#aggregate()函数按NAME\_2属性进行聚合

plot(p)

plot(pa,add=T,col=rainbow(12))#出图结束