Subject: R勉強会

Date: 2020/6/28

Name: Osamu Nishiura

目次

・オブジェクト

・型

・データフレーム取り扱い

・GDXファイル読み込み

・R studioの便利な機能

・オブジェクト

Rで取り扱う対象(?) のことをオブジェクトと呼ぶ。数値や文字列、それらを要素としたベクトル、関数、データなど様々なものが入る箱のようなものである。文字で説明するわかりにくいが、例を見ればすぐにわかるはず…

例：a <- 2

このaがオブジェクトである。aとかbとかは自分で勝手に設定してよいが、使用してはいけない文字列がいくつかあるので注意が必要である。

・型

　Rのオブジェクトはデータ型と属性の二つで構成される。データ型は実数、複素数、文字列などの要素、属性は成分の名前や次元、所属するクラスなどオブジェクト自体の特徴のことである。ここではわかりやすさを優先してオブジェクトの要素とオブジェクトの属性という分け方をする。

要素

・double 実数 3.1415

・integer 整数 3

・complex 複素数 1+1i

・logical 論理型 TRUE, FALSE

・character 文字列 “Kyoto”

・factor 因子型 Kyoto

上5つは、見ればわかるものばかりである。しかし、要素っぽい扱いをされるものの中に「因子型」というややこしいものが存在する。

> x<-c("dog","cat","cat","dog","rabbit","cat") ここまでは文字列

z<-factor(x,levels=c("dog","cat","rabbit"))　因子型に変換　levelsで順番を決められる。

> z

[1] dog cat cat dog rabbit cat

Levels: dog cat rabbit

要素がグループ化されている。Factorのかわりにorderedを使うと要素の大小関係も指定できる。因子型を用いることでソートや大小比較などができて便利になる。一方で、文字列型と混同しやすくエラーの原因となることも…

属性

・vector ベクトル

・matrix 行列

・array 配列

・list リスト

・data.frame データフレーム

上三つは構成する要素が1種類のオブジェクトである。リストはすべての属性(リストも含め)のオブジェクトを要素とできる。データフレームは要素が複数種類のベクトルで構成される2次元のオブジェクトである。GAMSなどの計算結果を集計、図化するにあたってはもっぱらデータフレームを扱うこととなると思われる。

・データフレームの取り扱い

　データフレームはベクトルをいくつか並べることでも作成することができる。

a <- c("apple","orange","grape","apple","orange")

b <- c("ringo","mikan","budou","ringo","mikan")

c <- c(10,12,3,8,15)

d <- c(T,T,T,F,T)

df <- data.frame(name=a,japanese=b,weight=c,damage=d)

View(df)



　ただ、おそらく実用上はGDXファイルやエクセルファイル、csvファイルなどを読み込むことになるはず。今回は扱いが容易なcsvファイルを読み込む。

df\_r <- read.csv("../data/rice.csv")

df\_o <- read.csv("../data/onion.csv")

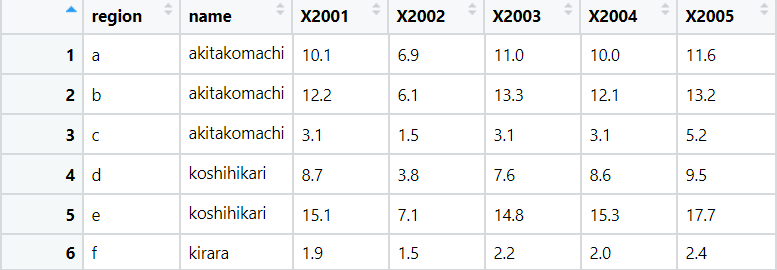
データを読み込めたところでそのデータをチェックする。

View(df\_r) オブジェクトの要素の確認

str(df\_r)　それぞれの列の型の確認

summary(df\_r)　それぞれの列の情報の確認

attributes(df\_r)　オブジェクトの型の確認



読み込むファイルと比べるとわかるが年にXがくっついている。これはデータフレームを扱う上で不都合を生じにくくするために、勝手にくっつけられたものである。また、nameの列を確認するとデータの型がfactorとなっている。

特別何かパッケージを入れなくてもデータフレームの読み込みや操作は可能であるが、上記のような問題が発生したり、プログラムが長くなったり読みにくくなったりしてしまう。そこで、データフレームを扱うのに便利なパッケージを導入する。

install.packages("tidyverse")

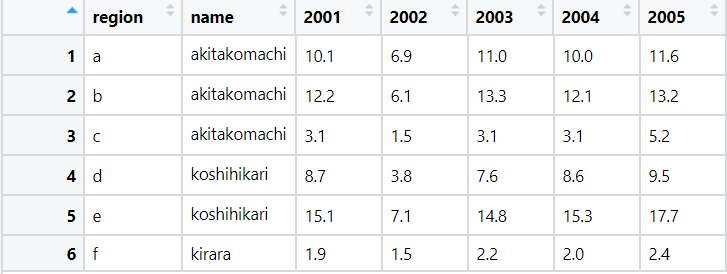
library(tidyverse)

tidyverseパッケージ一つで、ggplot2、dplyr、tidyr、readr、purrr、tibble、stringr、forcatsといったデータを扱うのに便利なパッケージ群を一括でインストールできる。これらのうち、今回はdplyr、tidyr、readrを使用する。

　readrではreadcsvで起きるような暗黙の型変換などが起きないので便利である。

df\_r <- readr("../data/rice.csv")

View(df\_r)



filter select %>%

　現在のデータは小さいので必要性を感じることはないかもしれないが、大きなデータを扱うためにはデータの抽出が不可欠である。そこで、filter関数(行の抽出)とselect関数(列の抽出)を使用する。例えば、地域aとbの2003年から2005年のみのデータをつくるには、

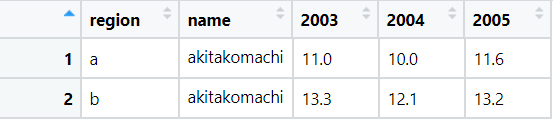
View(select(filter(df\_r,region=="a"|region==”b”),"region","name","2003","2004","2005"))

こんなかんじで入れ子にしてもできなくはないが、処理の数が増えるとわかりにくい。途中で新しい変数を作ったりしてもいいが、変数が増えるとややこしかったりメモリを食ったりよろしくない。そこで、パイプという便利なものを使用する。これは先に処理したオブジェクトを第一引数として次の処理に渡すもので、上記の問題を解決する。

filter(df\_r,region=="a"|region==”b”)%>%

select("region","name","2003","2004","2005")%>%

View()



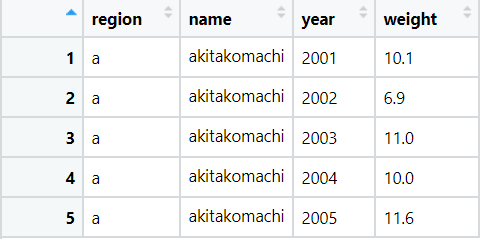
gather

よく見る表は上のような横型が多いが、このままではR上では扱いにくい。この表を縦型に変換する。全部出すと縦に長すぎるので地域aを抽出して表示する。

df\_r <- gather(df\_r,-c('region','name'), key='year', value='weight')

filter(df\_r,region=="a")%>%

View()



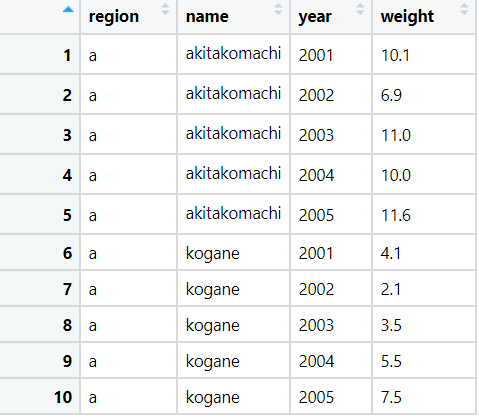
rbind

練習用のプログラムのデータには、米と玉ねぎについての2つのデータがある。これをrbind関数を使って一つに結合してみる。これもaのみ抽出

df\_b <- rbind(df\_r,df\_o)

filter(df\_b) %>%

View()



mutate

おそらく玉ねぎの品種名は一般的でないので品種名をやめて簡単に「米」と「玉ねぎ」と変換したい。そこでmutate関数を使う。

df\_b <- mutate(df\_b, name=recode(name,"akitakomachi"="rice",

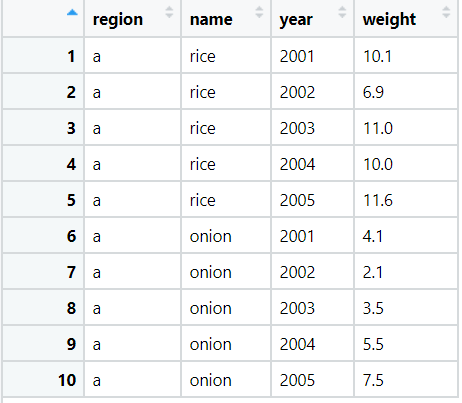
"koshihikari"="rice",

"kirara"="rice",

"kogane"="onion",

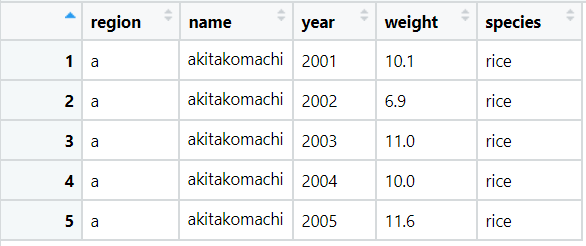
"kohaku"="onion"

))



Mutate関数はなかなか便利で新しく列を作成したりもできる。

df\_r <- mutate(df\_r,species="rice")



=の後ろには関数なども入れられて、行ごとの集計や単位の変更などなかなか便利

spread merge

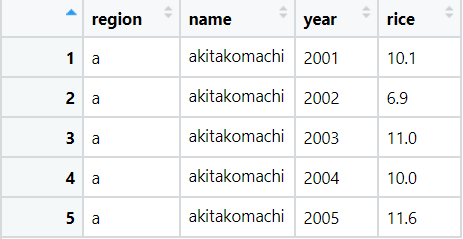
欠損データの扱いについて、玉ねぎのデータは地域eとfがかけている。単純にデータを縦に結合すると「かけている」という情報が失われてしまう。そこで一行にある地域、年において米と玉ねぎの生産量が示されるように二つのデータを結合する。まず、横方向に品目を並べることで列名を品目と一致させる。

df\_r <- spread(df\_r, key = species, value = weight)

df\_o <- spread(df\_o, key = species, value = weight)

filter(df\_r,region=="a")%>%

View()

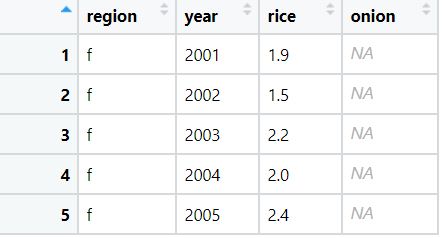


そして関数mergeを使用する。この関数は一致する地域年のところにデータを結合してくれ、データがない場合にはNA(データなし)としてくれる。

df\_a <- merge(df\_r[,c(1,3,4)],df\_o[,c(1,3,4)],all=T)

filter(df\_a,region=="f")%>%

View()



NA取り扱い

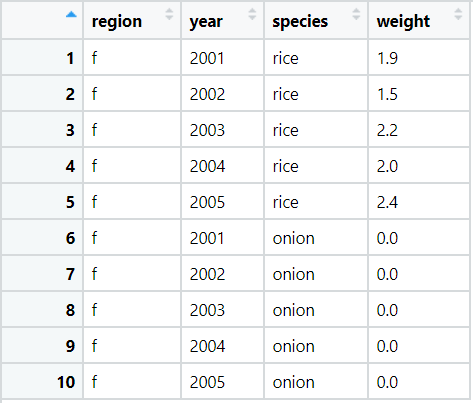
NAの箇所についてna.omit(df\_a)などで無視することもできるが、この場合は単ねぎを生産していないだけと思われるのでNAに0を代入する。

df\_a[is.na(df\_a)]<-0

これをもう一度縦型に変換すると地域eとfについても玉ねぎの生産量が入ったデータにすることができる。データを縦にしたり横にしたりすると計算やNaなどの扱いが容易になることがある。

df\_a <- gather(df\_a,-c('region','year'), key='species', value='weight')

View(filte(df\_a, region==”f”))



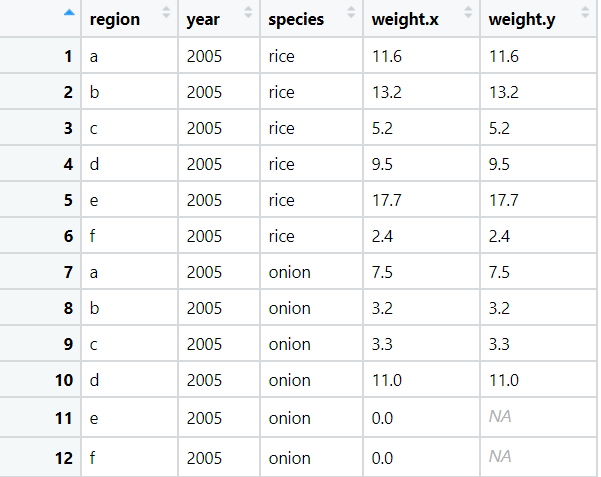
join

ここまでプログラムを進めると、コメと玉ねぎのデータを結合したデータがdf\_aとdf\_bの二種類ある。これらのデータの違いは米玉ねぎの列名と地域eとfの玉ねぎのデータがあるかな以下である。特に意味はないがこの二つのデータを結合してみる。データの結合をする際に列名が違うとmergeではうまく扱えない。列名を変更してもいいがdplyrのjoin関数を使うと簡単に結合できる。fulljoinではデータがない部分にはNAを入れてくれる。join系の関数はデータが存在する部分のみを返したり、逆に存在しない部分のみを返したりにも使える。

full\_join(df\_a,df\_b,by=c("species" = "name","region","year"))%>%

filter(year=="2005")%>%

View()



for if

最後にforとifの使い、すべての地域の2001年から2005年の合計生産額を計算する。ここで米は10円玉ねぎは20円として合計する。

s<-0

for (i in 1:nrow(df\_a)){

if(df\_a[i,3] == "rice"){

s <- s + df\_a[i,4]\*10

} else {

s <- s + df\_a[i,4]\*20

}

}

s

[1] 4199

データの集計に関しては関数がいろいろあり、うまく使えばコードがとても短く済む。しかし、どうしても見つからないときや複雑な処理をさせようと思うとどこかで不都合が生じることもある。そうしたときの最終手段としてifやforを使って力業でやってしまおう。

・GDXファイルの読み込み

　GAMSによる計算結果をRで扱うためにはGDXRRWというパッケージを使用する必要がある。

・Rstudioの便利な機能

Rstudioには便利な機能が多く存在する。

今回のRの勉強会では大雑把にデータフレームなどの取り扱いを体験してもらうことを目的とした。実際にデータを扱う際には、これら以外の関数も様々に組み合わせることになると思われる。RはGAMSと比べると超メジャーなので調べればたくさんの情報が見つかるはずである。私がよく参考にするサイトをいくつか掲載しておく。

統計解析フリーソフト R の備忘録頁 ver.3.1

<http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html>

データハンドリング入門　<https://kazutan.github.io/kazutanR/hands_on_170730/index.html>

dplyrを使いこなす！基礎編

<https://qiita.com/matsuou1/items/e995da273e3108e2338e>