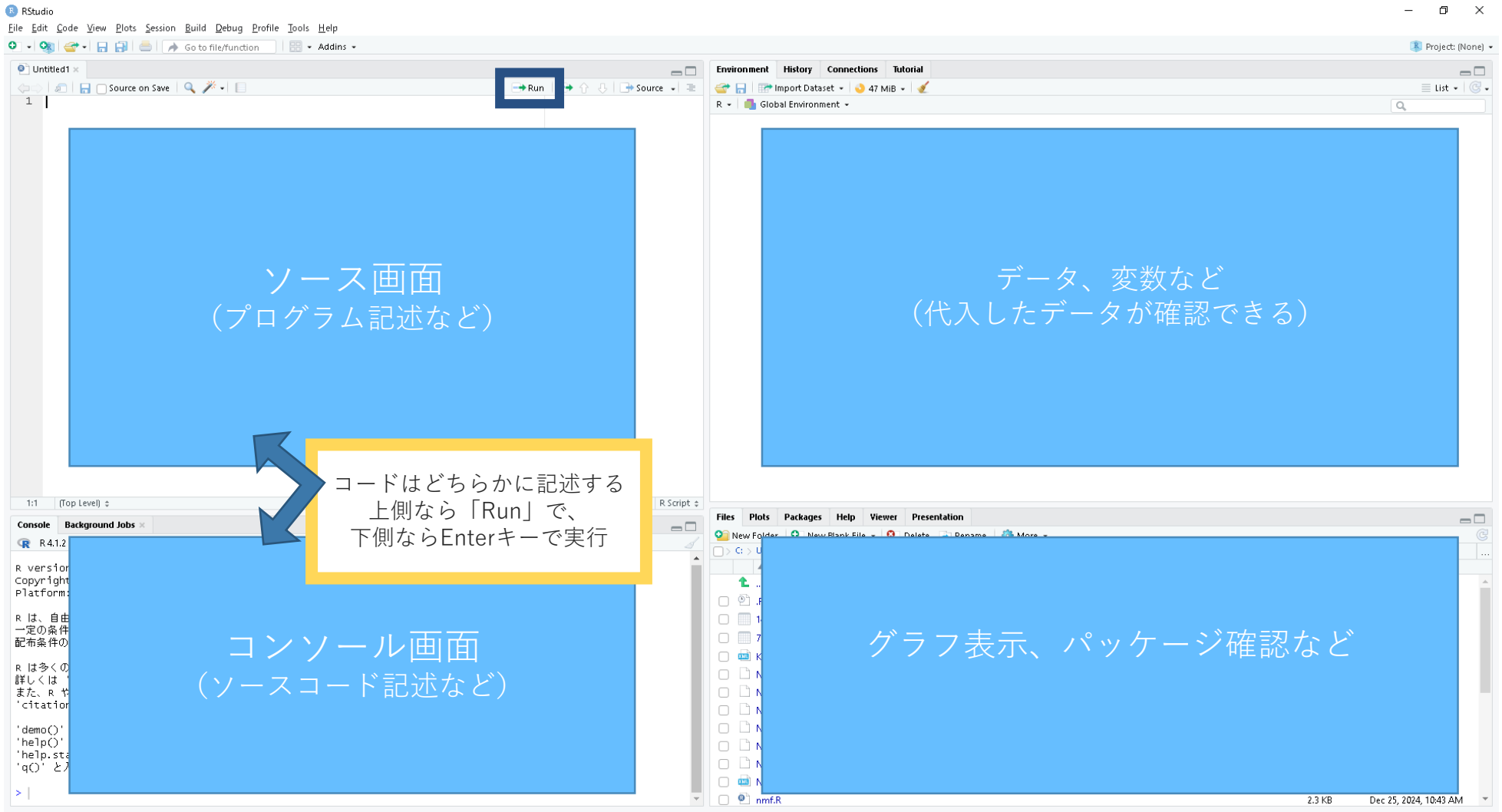
Rを使った分析
スクリプト解説

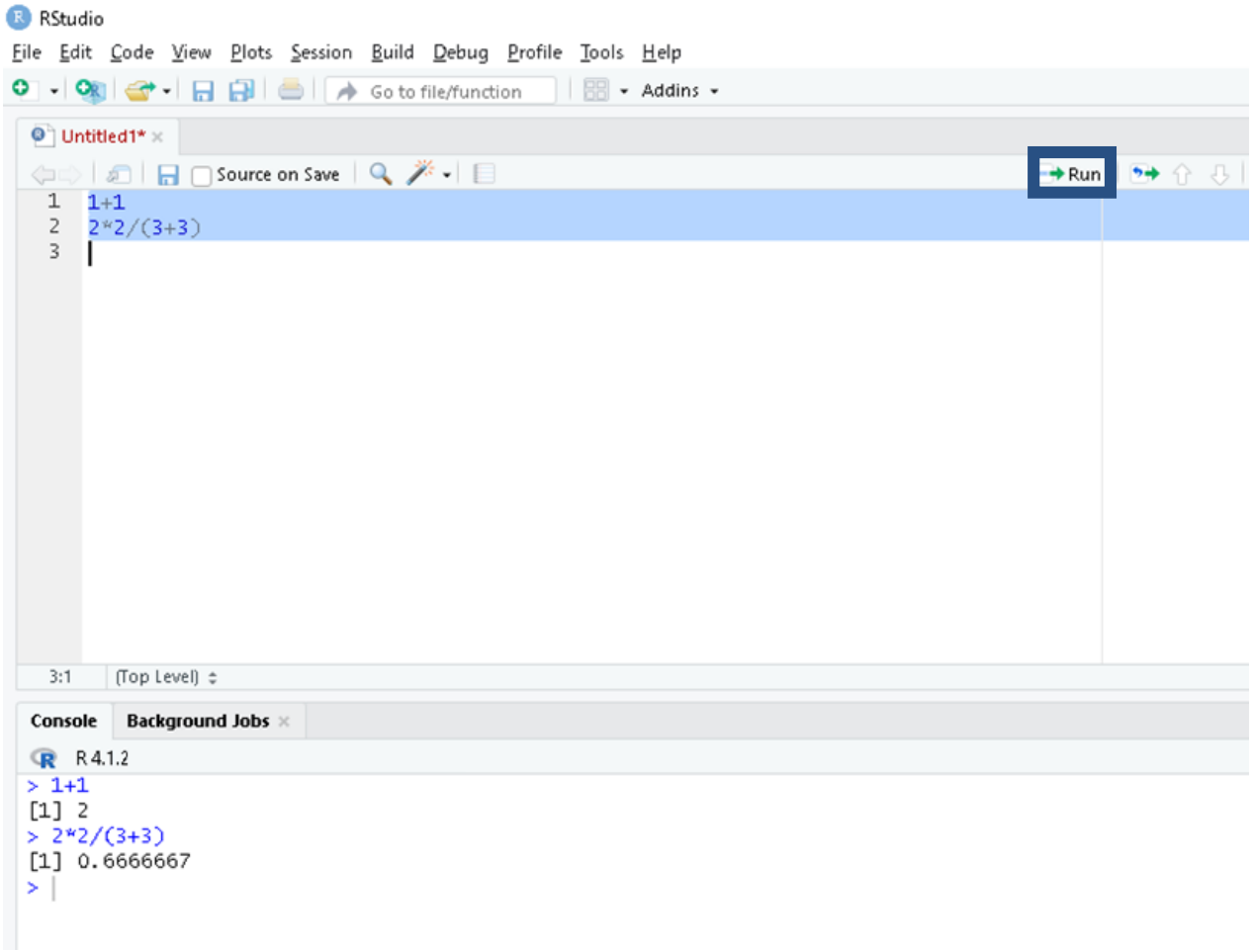
目次

Rstudioの画面構成



RStudioの基本操作＞計算式の入力

✓Rにコードを入力する際は半角英数で行います。



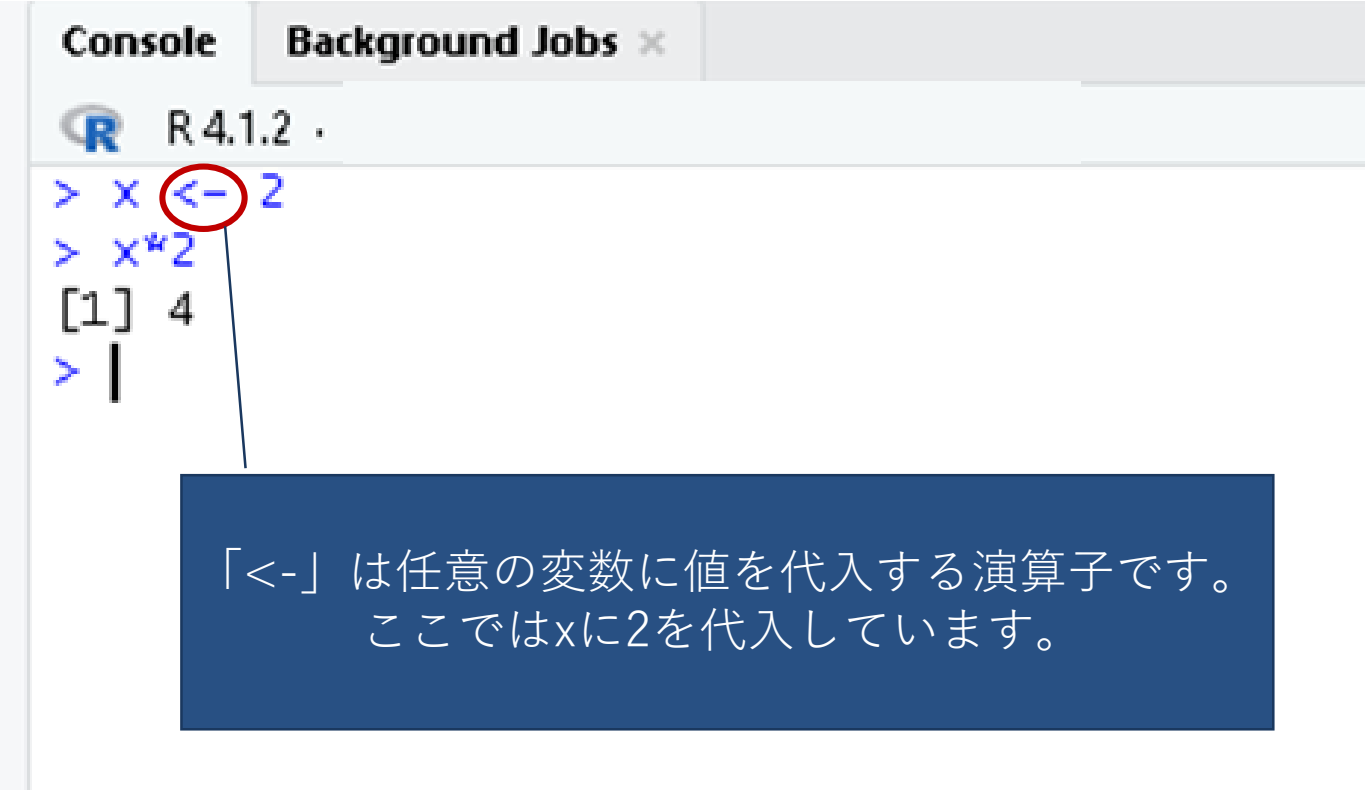
左上の画面に、
1+1
2*2/(3+3)
と入力し、入力部分を選択して「Run」を押してみました。

Console画面に、結果が表示されました。

✓ 複数行をまとめて実行する場合は、実行する部分を全選択してから実行します。

記号	演算
+	足し算
-	引き算
*	掛け算
/	割り算

RStudioの基本操作＞計算式の入力



次はConsole画面に、
x <- 2
x*2
と入力し、Enterキーを押してみました。

Console画面に、
[1]4
と表示されました。

この式では、xに2を代入し、xに2を掛けている式なので、2×2=4となるためです。

任意の変数に値を代入することで、同じ数字を何度も入力する手間が省けます。

Rstudioの基本操作＞関数の使用

基本パッケージにある主な関数	
mean()	平均値を計算
sd()	標準偏差を計算
cor()	相関係数を計算
data.frame()	データフレームを作成
plot()	基本的なプロット（グラフや図）を作成

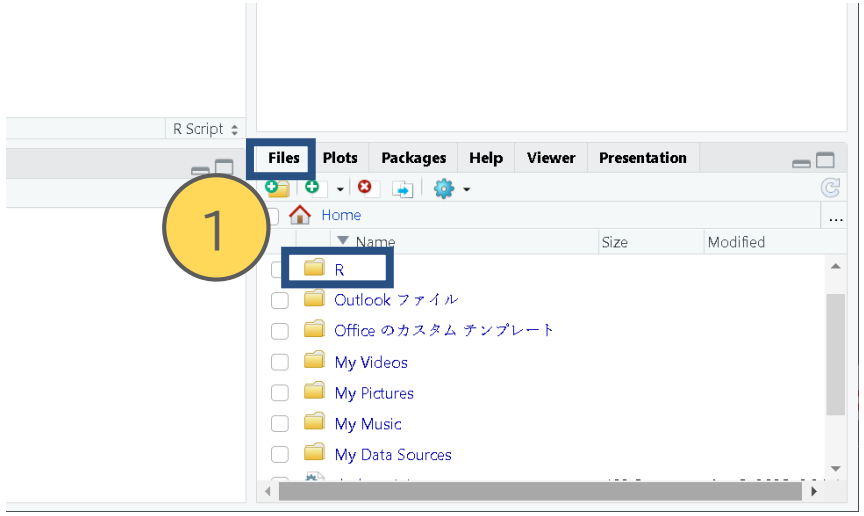
次はConsole画面に、
```
x <- 1
y <- 2
z <- 3
sum(x,y,z)
```
と入力し、Enterキーを押してみました。

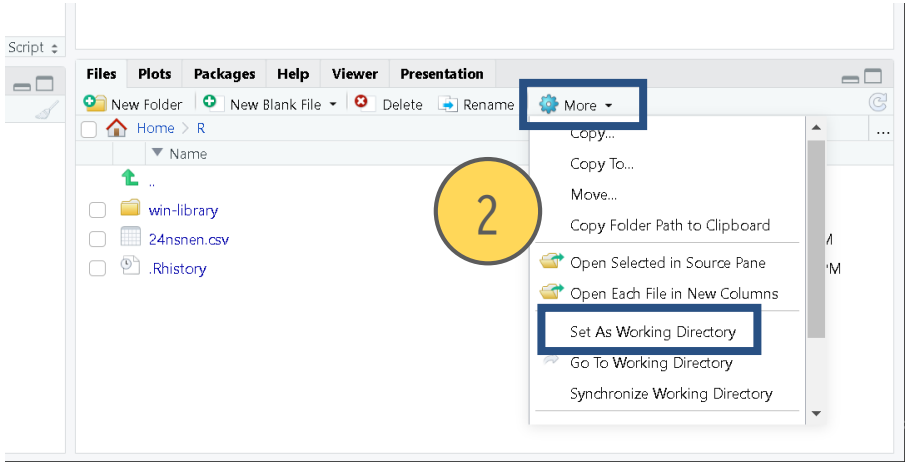
Console画面に、
```
[1] 6
```
と表示されました。

sum(x,y,z)では、
sum関数を使ってx,y,zを足し算しています。
sum(x,y,z)=1+2+3=6となるため、結果も6と表示されています。

Rには様々な関数があり、これらの関数を利用することで複雑な計算や処理を素早く行うことができます。

RStudioの設定＜データの読み込み①作業ディレクトリの指定





✓ Rでデータファイルを読み込む際に参照するフォルダ(ディレクトリ)を指定

Rでファイルを読み込む場合、読み込むファイルのフォルダと名前を指定する必要があります。
ファイルを読み込む度にフォルダを指定するのは大変なので、始めに参照するフォルダ（作業ディレクトリ）を指定しておきます。

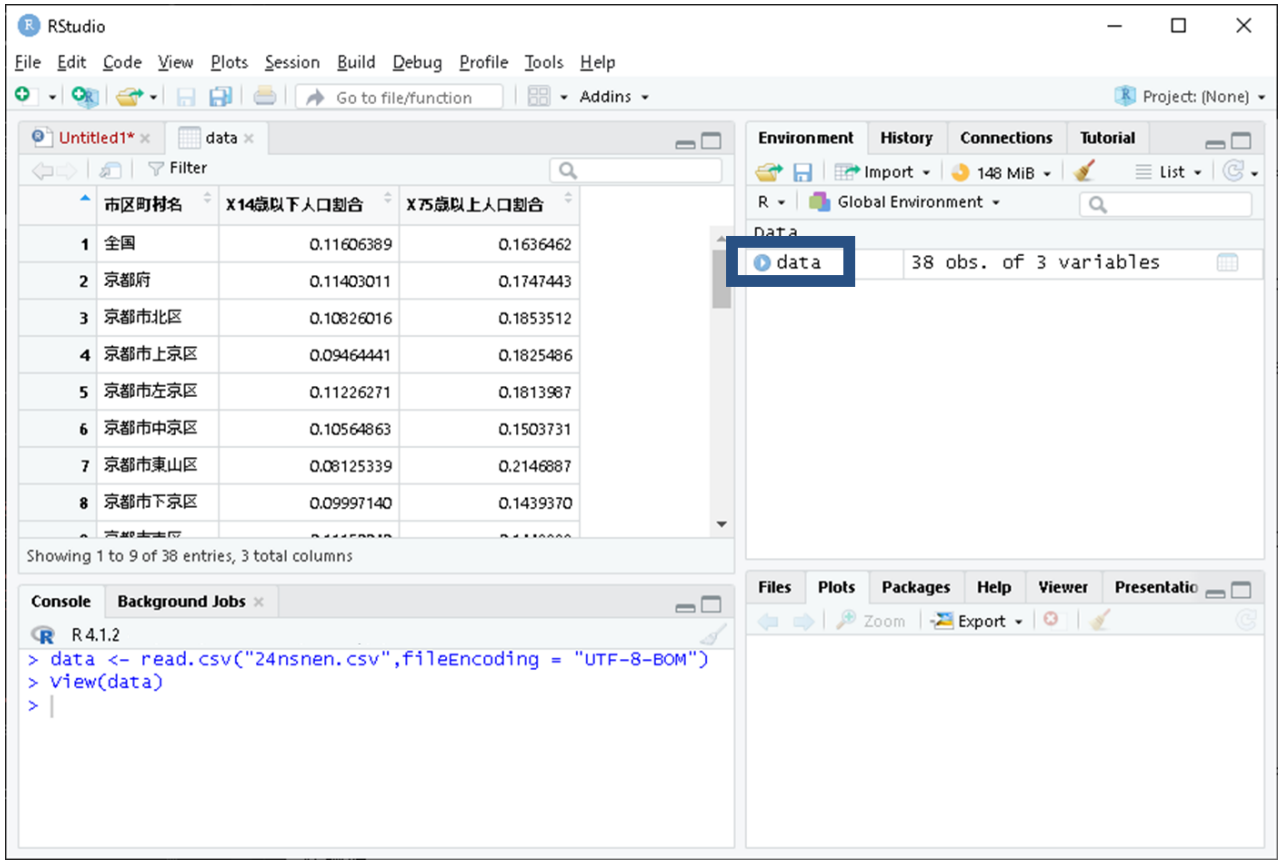
①右下画面の「Files」タブから任意のフォルダを選択（ここではユーザのドキュメント内の「R」というフォルダ）し、クリック。
②「More」＞「Set As Working Directory」を選択。

Console画面に、
setwd("~/R")
と表示され、作業ディレクトリが指定できました。

読み込むデータファイル（ここでは24nsnen.csv）はこの作業ディレクトリに保存しておきます。
また、Rで保存するファイルも特に指定しなければこの作業ディレクトリに保存されます。

Console画面に
> getwd()
と入力することで現在の作業ディレクトリが確認できます。

Rstudioの基本操作＞データの読み込み②



次はConsole画面に、
`data <- read.csv("24nsnen.csv")`
と入力し、Enterキーを押してみます。

右上の画面に、dataと表示されました。
dataをクリックすると、左上の画面にデータが表示されました。

`data <- read.csv("24nsnen.csv.csv")`は、
`read.csv("ファイル名")`で読み込んだデータを、
dataに格納して、dataを見る、という操作です。

読み込んだデータを見ると、2，3列目のラベル行の1文字目にXが入力されていることが分かります。
これはラベルなどの変数が数字から始まるときに自動で入力されるためです。

エラーになるときは、以下も試してみてください。
またOSのバージョンによっては、文字化けすることがありますので、別の文字コードも試してみてください。
`data <- read.csv("24nsnen.csv",fileEncoding = "UTF-8-BOM")`

この解説の見方の説明

- 灰色のボックスにはRのスクリプトを記載しています。
- #以下はコメント（注釈）として、補足を記載しています。（#以下のコメントは入力してもコンピュータに無視され、影響がありません。入力不要です。）
- 文頭の「>」は、Consoleにコードを入力する際に自動で挿入される記号です。
  「>」は入力する必要がありませんが、ここでは複数行にわたるスクリプトを1つのスクリプトとして分かりやすくするために記載しています。

```
> data <- read.csv("24nsnen.csv") # データの読み込み
> rownames(data) <- data[,1] # 1列目を行名にする
```

- data <- read.csv(“24nsnen.csv”) :「24nsnen.csv」という名前のファイルを読み込んで「data」に格納します。
- rownames(data) <- data[,1] :dataの1列目(市区町村名)を行名(rownames)にします。

- 白いボックスにはRのスクリプトごとの解説を記載しています。
- 該当のスクリプトを記載した後、「：」以下には関数や引数（「,」で区切られている部分）の設定の解説をしています。

Rを使った分析＞データの読み込みと整頓

```
> data <- read.csv(“24nsnen.csv”)  # データの読み込み
> rownames(data) <- data[,1] # 1列目を行名にする
> data <- data[,-1] # 1列目を削除する
> head(data) # 先頭から6行のデータを表示する
```

- data <- read.csv(“24nsnen.csv”) :「24nsnen.csv」という名前のファイルを読み込んで「data」に格納します。
- rownames(data) <- data[,1] :dataの1列目（市区町村名）を行名（rownames）にします。
- data <- data[,-1]:市区町村名は行名になったため、不要になったdataの1列目を削除します。
- head(data):dataの先頭から6行のデータを表示して、データの整頓を確認します。

```
> data <- read.csv("24nsnen.csv",fileEncoding = "UTF-8-BOM")
> rownames(data) <- data[,1]
> data <- data[,-1]
> head(data)
           ×14歳以下人口割合 ×75歳以上人口割合
全国              0.11606389       0.1636462
京都府            0.11403011       0.1747443
京都市北区         0.10826016       0.1853512
京都市上京区        0.09464441       0.1825486
京都市左京区        0.11226271       0.1813987
京都市中京区        0.10564863       0.1503731
>
```

4行目まで実行したConsole画面です。

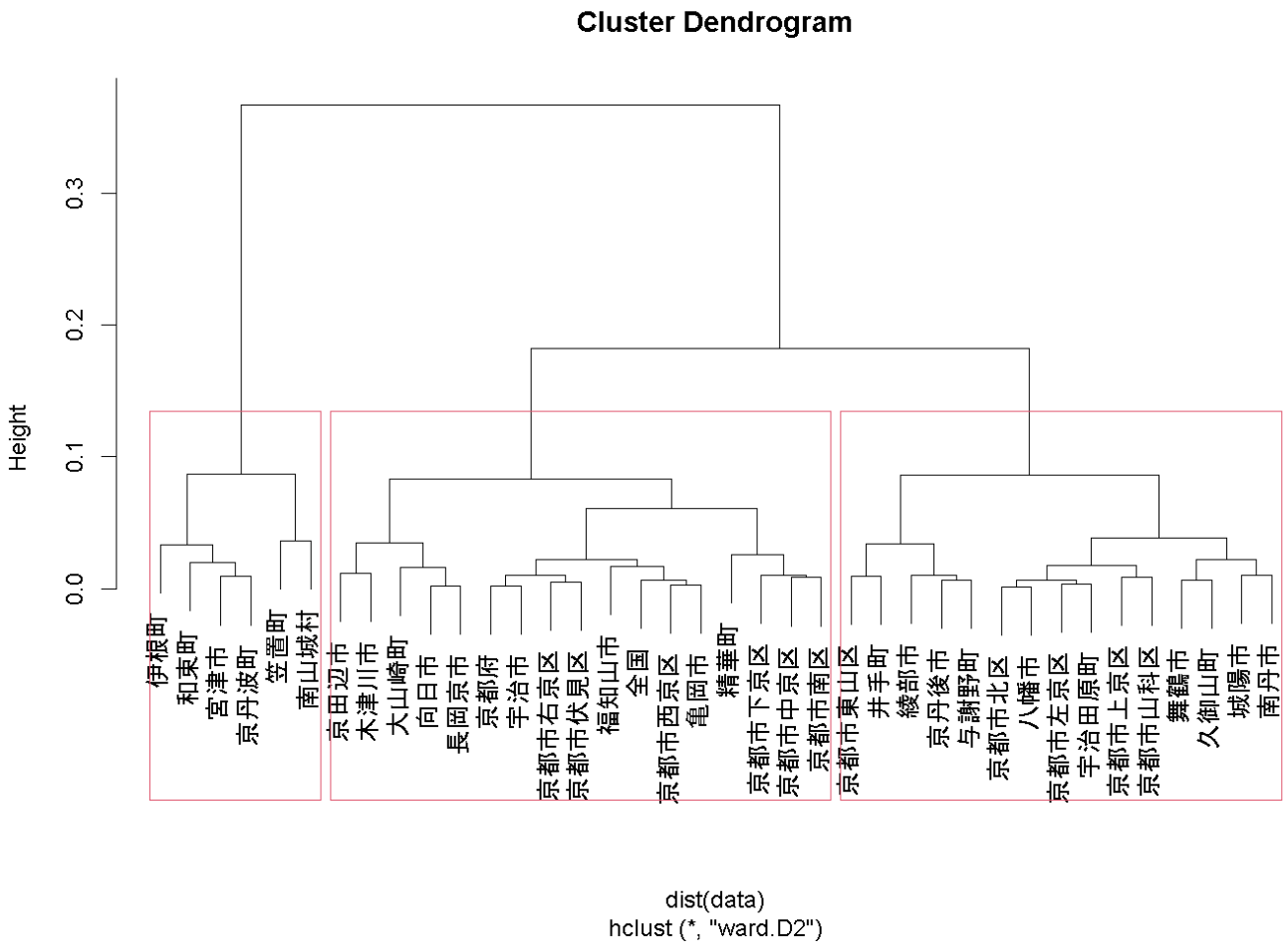
head関数の結果は黒字で表示されています。

Rを使った分析＞階層型クラスター分析と可視化

```
> res.hclust <- hclust(dist(data),method="ward.D2") # ウォード法による階層型クラスター分析
> plot(res.hclust) # デンドログラムを描く
> K <- 3
> res.rect <- rect.hclust(res.hclust,k=K) # K個のクラスターに分類する
```

- `res.hclust <- hclust(dist(data),method="ward.D2")`: `dist(data)`でdataの距離行列を計算し、この距離行列を`hclust`関数（ウォード法）で階層型クラスター分析しています。分析の実行結果は`res.hclust`に格納しています。
- `plot(res.hclust)`: plot関数で、階層型クラスター分析結果のデンドログラムを作成します。
- `K <- 3`: Kに3を代入します。
- `res.rect <- rect.hclust(res.hclust,k=K)`: `rect.hclust`関数で、階層型クラスター分析結果をK個のクラスターに分類して、デンドログラム上にクラスターの境界を四角形で表示します。ここではK=3なので、3つのクラスターに分類しています。

Rを使った分析＞デンドログラムの確認①



✓デンドログラムが表示されました。

トーナメント表のように見えますが、枝分かれしている部分が木のようにも見えるので、「樹形図」とも呼ばれます。

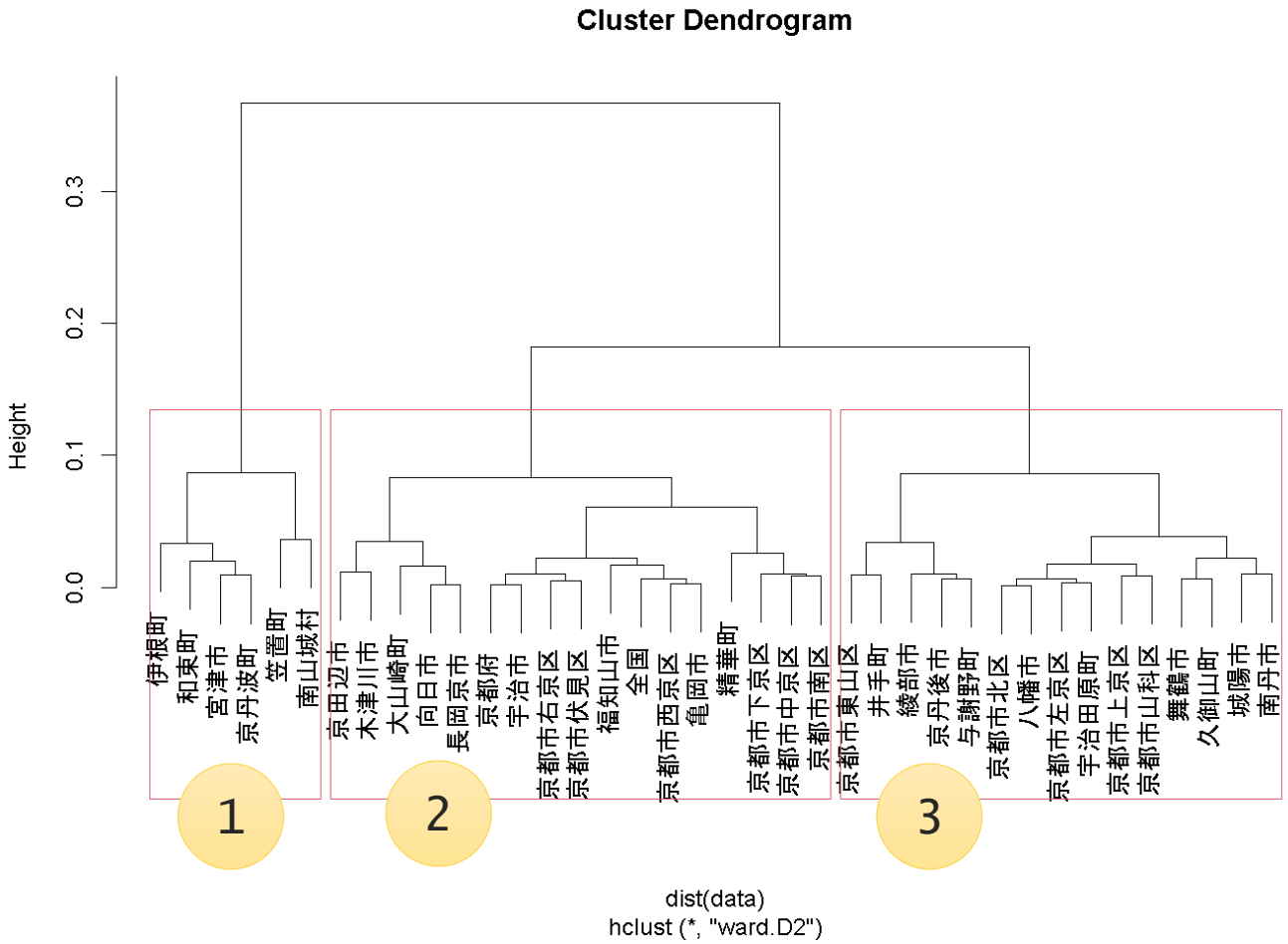
【デンドログラムの構造】

- 葉（リーフ）：図の一番下にある部分で、各データポイント（ここでは市区町村）を表す。
- 枝（ブランチ）：データポイントがグループ化される過程を示す線。似ているデータ同士が枝でつながれる。
- 結合点（ノード）：枝が交わる点で、データがどの段階でグループ化されたかを示す。
  結合点の高さは、データ同士の距離を表す。

【デンドログラムの見かた】

葉からスタートし、どのデータがどのようにグループ化されているかを確認します。
枝を辿っていくと、データがどの段階で結合されているか分かります。結合点の高さが高いほど、データ同士の距離が大きいことを意味し、結合点が低いほどデータ同士が似ていることを意味します。

Rを使った分析＞デンドログラムの確認②



✓ 結果の確認

今回作成したデンドログラムは、2024年の14歳以下人口割合と75歳以上人口割合の2つのデータでクラスタリングした結果です。

赤線で囲まれているのが、rect.hclust関数で分類した3つのクラスターです。
（後の説明の便宜上、1，2，3としておきます）

最初の結合点の高さ（データ間の距離）に大きな差はありませんでしたが、一番距離が近いのは京都市北区と八幡市の0.017で、一番距離が遠いのは木津川市と笠置町の0.225でした。
また、京都府の平均と一番似ているのは宇治市で、全国の平均と一番似ているのは京都市西京区と亀岡市のグループでした。

読み込むデータファイルを変えれば、他のデータでも同様のクラスタリングができます。
是非色んなデータで試してみてください。

Rを使った分析＞グラフの作成①

```
> mycol <- rep(1,nrow(data))
> for(k in 1:K) mycol[res.rect[[k]]] <- k # クラスター番号を色に設定する
```

作成するグラフが見やすくなるよう、データの各行に対して、クラスター番号を色として設定します。

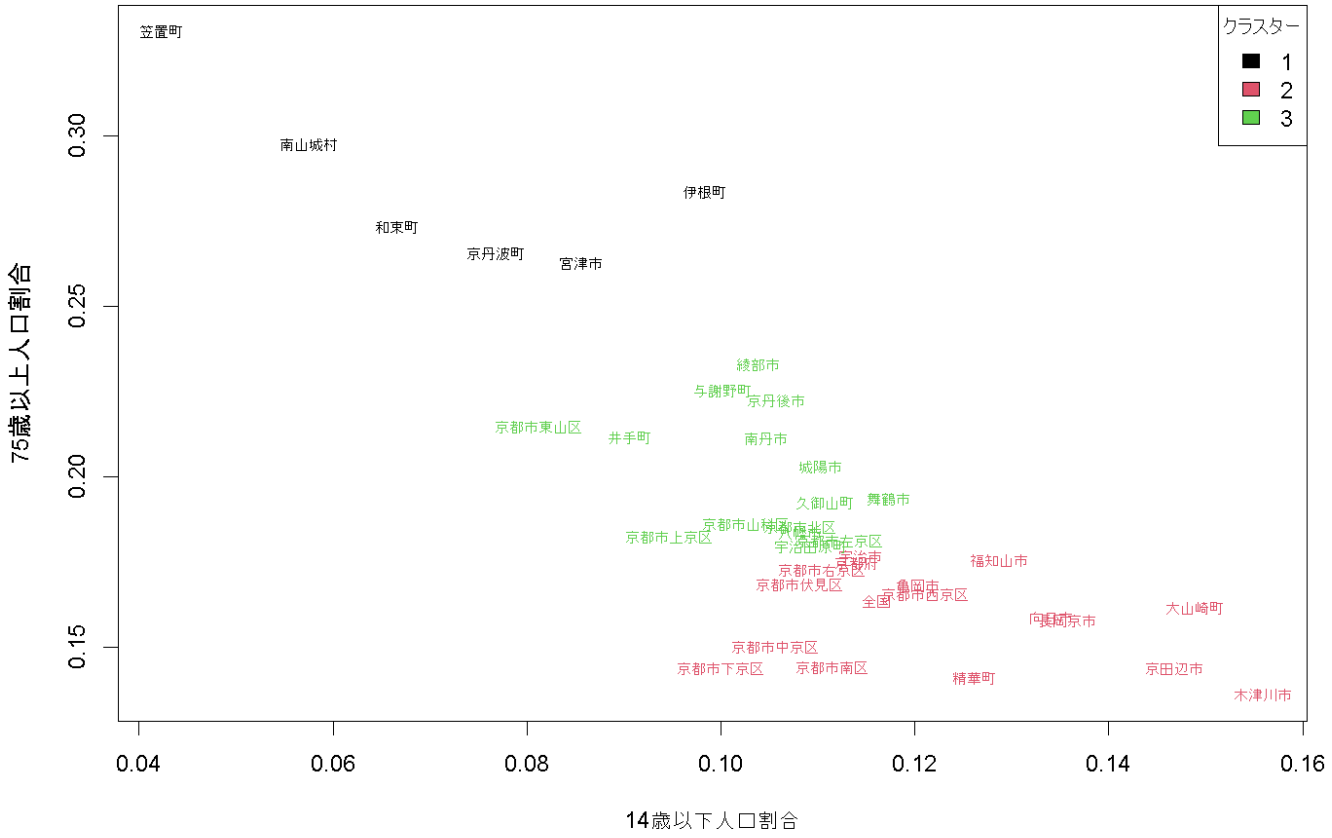
- `mycol <- rep(1,nrow(data))`: dataの行数と同じ長さのベクトルを作成し、すべての要素を1に設定し、mycolに格納します。
- `for(k in 1:K)mycol[res.rect[[k]]] <- k`: `for(k in 1:K)`で1からKまでのクラスタ番号についての指示を実行することを先に宣言します。
  `res.rect [[k]]`は、K個(ここでは3個)に分類したクラスターのうち、クラスターkに属するデータのインデックスを含むリストです。(例えば、クラスター1なら6市町村の行番号のリスト)
  `mycol[res.rect[[k]]] <- k`は、クラスターkに属するデータのインデックスに対して、mycolベクトルの対応する要素をkに設定します。これにより、各データポイントにクラスター番号が割り当てられます。(各市区町村に、グループ番号(ここでは色)を割り振っています)

Rを使った分析＞グラフの作成②

```
> plot(data,type="n",xlab="14歳以下人口割合",ylab="75歳以上人口割合") # 散布図を描く
> text(data,rownames(data),cex=0.6,col=mycol) # クラスター番号を色としたラベルを描く
> legend("topright",legend=1:K,fill=1:K,title="クラスター") # 凡例を付ける
```

- plot(data,type=“n”,xlab=“14歳以下人口割合”,ylab=“75歳以上人口割合”)： plot関数でdataのグラフを作成します。typeはグラフの種類を指定しています。”n”はなにもプロットせず、後で追加するプロットのためにデータポイントのプロット領域のみを設定するものです。
  xlabは横軸のラベルを、ylabでは縦軸のラベルを設定しています。
- text(data,rownames(data),cex=0.6,col=mycol) ： text関数で、データポイントの一にテキストラベルを追加してます。dataでテキストを配置する位置を、rownames(data)で各データポイントに表示するラベル(ここでは行名(市区町村名))を、cexでテキストのサイズを、colでテキストの色(先にグループ番号ごとに割り振った色)を指定しています。
- legend(“topright”,legend=1:K,fill=1:K,title=“クラスター”)： legend()で凡例の設定をします。
  “topright”で凡例の位置(ここでは右上)を、legend=1:Kで凡例を表示する範囲を、fil=1:kで凡例の各項目に対して色を、titleで凡例名を設定しています。

Rを使った分析＞グラフの確認①



✓グラフが表示されました。

このグラフは、数値的にはExcelで作成したグラフと同じものです。
プロットの点を市区町村名の表示に変え、更に市区町村名の表示色を階層型クラスター分析で分類した3つのグループごとに分けています。

それぞれクラスター１が黒色、クラスター2が赤色、クラスター3が黄緑色で表示されています。

こうすることで、14歳以下人口割合と75歳以上人口割合の実際のデータ分布と、クラスタリング結果が一目で確認できます。

Rを使った分析＞図やグラフを保存する

- PlotパネルのExportボタンや表示される図を右クリックすることで保存できますが、関数を使うことで図やグラフを直接作業ディレクトリに保存することもできます。ここではPNGファイルの保存形式について参考にご紹介します。
- 最初にpng関数でファイル名を指定し、plot関数でグラフを描画した後、dev.off()関数で終了する、という流れです。
- 具体的には、以下のようになります。下線部の関数で、グラフ作成の過程を挟む形です。

```
> png("hclust.png",width=1400,height=1000,pointsize=24)
> res.hclust <- hclust(dist(data),method="ward.D2")
> plot(res.hclust)
> K <- 3
> res.rect <- rect.hclust(res.hclust,k=K)
> dev.off()
```

- png(“hclust.png”,width=1400,height=1000,pointsize=24): png関数でPNGファイルに保存するためのデバイスを開きます。””でファイル名を、widthで横幅(ピクセル単位)を、heightで高さを、pointsizeで文字の大きさを指定しています。ファイル名の指定だけでも出力できます。
- dev.off(): dev.off関数でデバイスを閉じてファイルを保存します。作業ディレクトリにファイルが保存されました。