Rによる気象データサイエンス

榎本剛

2023-06-04

Table of contents

第 1 章

R による気象データサイエンス

この本では R を使って気象データの解析や可視化を学びます。本書の URL は https:

//www.dpac.dpri.kyoto-u.ac.jp/enomoto/rmetds です。

データサイエンスでは tidyverse でデータを処理し、それに含まれる ggplot2 で可視化することが標準となっており、インターネット上の情報には tidyverse に基づく解説が多いようです。しかし、tidyverse は初心者向きではない[*1]と考えますので、本書では base R を用います。

参考

参考となるサイトを挙げます。R と RStudio にはレポートや本を作る機能があるので、たくさんの書籍が作れらており、多くはウェブ上に無料で公開されています。R は 1 文字だけなので、検索する場合は複数のキーワードを使いましょう。

- 奥村晴彦先生統計・データ解析
- 竹中明夫さん R でプログラミング：データの一括処理とグラフ描き
- 増田耕一さん R による気象データの作図、シミュレーションなど
- RjpWiki
- 宋財泫さん・矢内勇生さん私たちの R
- 榎本 R を気象研究に使う

[*1] Matloff, N, 2013: Teaching R in a Kinder, Gentler, More Effective Manner.

第 2 章

はじめに

R は統計が得意なプログラミング言語で、きれいな図を作ることができます。

2.1　R をインストールする

早速 R をインストールしてみましょう。R のウェブサイトは https://www.r-project.org/ です。R はターミナル（mac のターミナルや Windows の Windows Terminal 等）から使うこともできますが、RStudio https://posit.co/download/rstudio-desktop/ から使うと便利です。

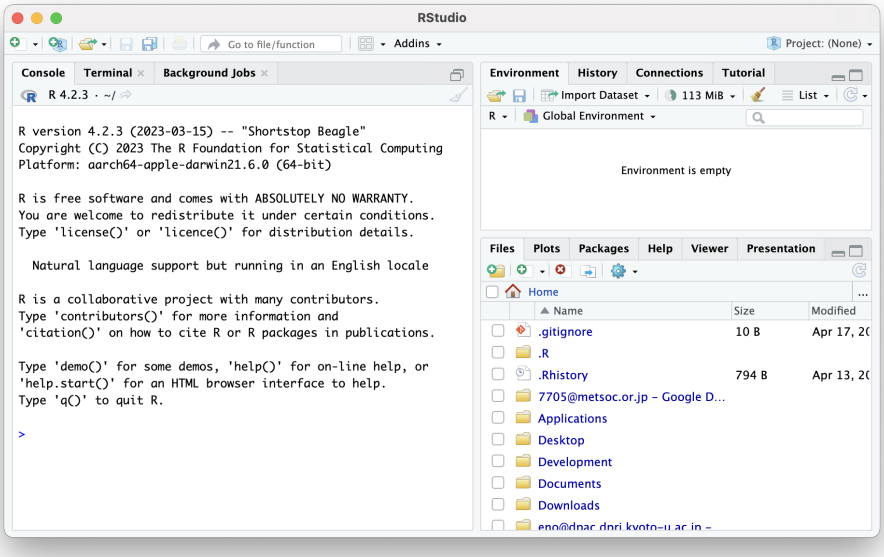


Figure2.1: RStudio

Download and Install R から自分の使っている OS のコンパイル済バイナリを取得してインストールしてください。

2.2　ヘルプを参照する

R のヘルプはコンソールで `help()` または ? を使うか、RStudio 右下の Help で検索できます。

```
help("mean")
```

2.3　計算をしてみよう

ターミナルで R を起動するか、RStudio を起動してください。

大気に含まれている主要な成分は、窒素、酸素、アルゴンです。これらの成分はよくかき混ぜられているので、乾燥空気の平均分子量は容積比を重みとした平均で計算できます。

Table2.1: 大気の主成分

成分	分子量	容積比 %
窒素	28	78
酸素	32	21
アルゴン	40	1

```
28 * 0.78 + 32 * 0.21 + 40 * 0.01
```

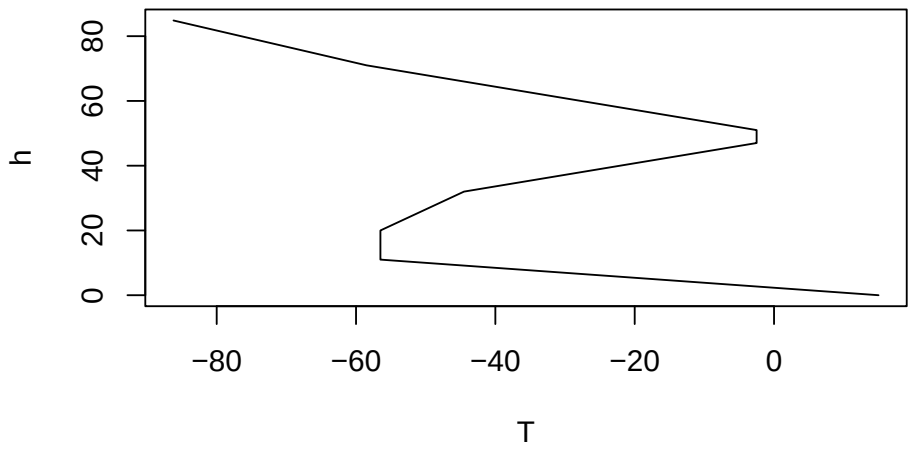
```
[1] 28.96
```

2.4　グラフを描いてみよう

標準的な気温の鉛直構造として、標準大気が定義されています。標準大気を縦軸をジオポテンシャル高度、横軸を気温として描画してみましょう。ジオポテンシャル高度, geopotential height とは、重力加速度を鉛直積分してジオイド面での重力加速度 $g_0 = 9.80065\,\mathrm{m\,s^{-2}}$ で割ったものです。

$$Z = \frac{1}{g_0}\int_0^z g(z')\mathrm{d}z' \tag{2.1}$$

```
T <- c(15.0, -56.5, -56.5, -44.5, -2.5, -2.5, -58.5, -86.2)
h <- c(0, 11, 20, 32, 47, 51, 71, 84.852)
plot(T, h, type="l")
```



<- は代入を表す演算子で左側の T や h に右側の値を与えています。c() はコンマで区切った値を繋げて（concatenate）ベクトルを作る函数です。ベクトルの要素は T[1] のように添字で参照できます。添字は Fortran 同様に 1 から始まります。Python では、-1 は最後の要素を意味しますが、R ではマイナスをつけると、その要素を取り除いたベクトルを返すので注意してください。plot() でグラフを描いています。引数として x、y の値である T と h、グラフの種類を折れ線グラフにする type="l" を与えています。

グラフはウィンドウの右下の Plots タブに表示されます。Export ボタンで画像や PDF に保存したり、クリップボードにコピーすることができます。

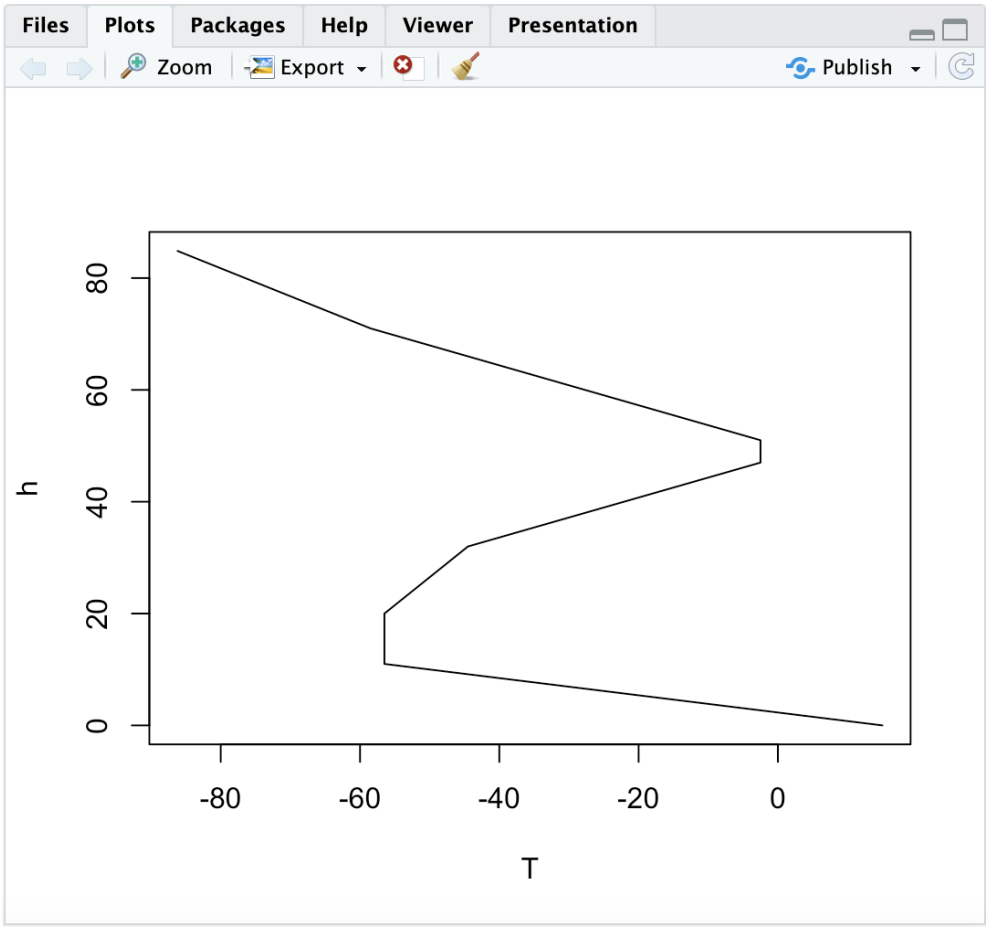


Figure2.2: 標準大気

2.5　プロジェクト

統計解析には、対象となるデータが必要であるだけでなく、解析のためのスクリプト、画像、報告書、プレゼンテーションなどが生成されます。これらのファイルは、一つのフォルダ（ディレクトリ）の中に集めておくと便利です。このディレクトリを作業ディレクトリと呼びます。R で作業ディレクトリを確認するには `getwd()`、設定するには `setwd(dir)` を使います。`dir` にはディレクトリのパスを `""` で括られた文字列で与えるか、文字列の入った変数を渡します。

RStudio を起動したときの作業ディレクトリは、Mac ではホームディレクトリ、Windows では「ドキュメント」フォルダです。Windows のドキュメントフォルダは、通常 OneDrive 上に設定されています。場所は Tools > Global Options で確認、設定できます。

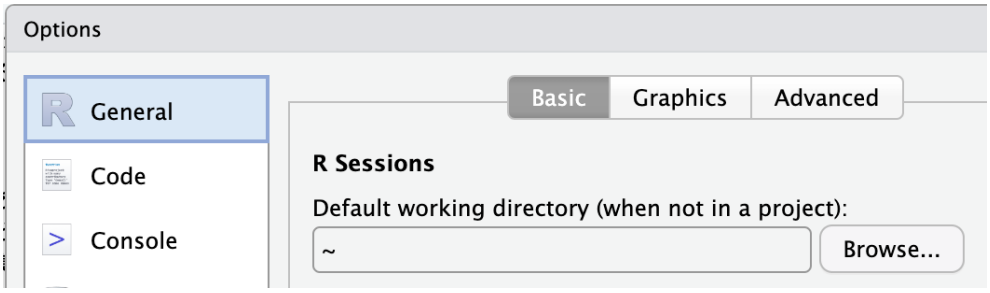


Figure2.3: Global Options

RStudio のプロジェクトを使うと、プロジェクトファイルのある場所が作業ディレクトリになります。プロジェクトを作るには、RStudio のタイトルバーのすぐ下のボタンのうち、左から 2 番目 +R と書いてある立方体をクリックします。新規プロジェクト、New directory を選択し、プロジェクトの名前と、作業ディレクトリを保存する上位のディレクトリを指定します。

第 3 章

CSV データ

CSV ファイルは `read.csv()` で読み込むことができます。この函数は、data.frame と呼ばれる、リストに似たオブジェクト返します。行や列に名前を付けて参照することができます。

3.1 測定データ

二酸化炭素の測定をして、1 列目が日付、2 列目が時刻、3 列目が測定値という CSV ファイルが得られたとします。

```
co2 <- read.csv("co2.csv", header=FALSE)
head(co2)
```

```
          V1       V2  V3
1 2024/5/17 17:41:15 633
2 2024/5/17 17:41:18 633
3 2024/5/17 17:41:27 637
4 2024/5/17 17:41:30 638
5 2024/5/17 17:41:33 639
6 2024/5/17 17:41:36 639
```

列に名前を付けましょう。

```
names(co2) <- c("date", "time", "co2")
```

名前を使って `co2$date` または `co2[["date"]]` とすると、列をベクトルとして取得できます。`co2$["date"]` は `date` を残した data.frame を返します。

日付と時刻を `POSIXct` に変換しておくと、時系列データを扱うときに便利です。`POSIXct` は R の日時オブジェクトの一つで、基準時刻からの秒数で表します。`ct` は calendar time を意味します。`POSIXlt` は local time で要素を整数のベクトルで表します。

```
co2$datetime <- as.POSIXct(paste(co2$date, co2$time))
head(co2)
```

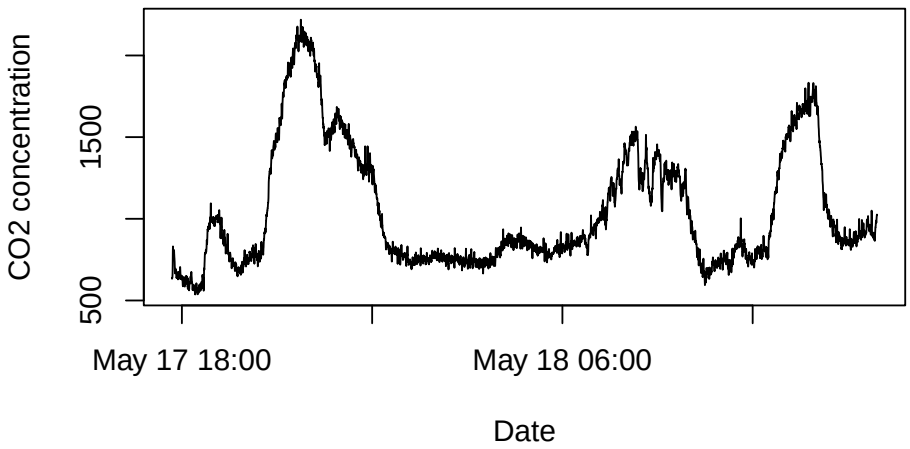
```
       date     time co2            datetime
1 2024/5/17 17:41:15 633 2024-05-17 17:41:15
2 2024/5/17 17:41:18 633 2024-05-17 17:41:18
3 2024/5/17 17:41:27 637 2024-05-17 17:41:27
4 2024/5/17 17:41:30 638 2024-05-17 17:41:30
5 2024/5/17 17:41:33 639 2024-05-17 17:41:33
6 2024/5/17 17:41:36 639 2024-05-17 17:41:36
```

不要な列は NULL を代入して削除します。

```
co2$date <- NULL
co2$time <- NULL
```

時間変化をプロットしてみましょう。

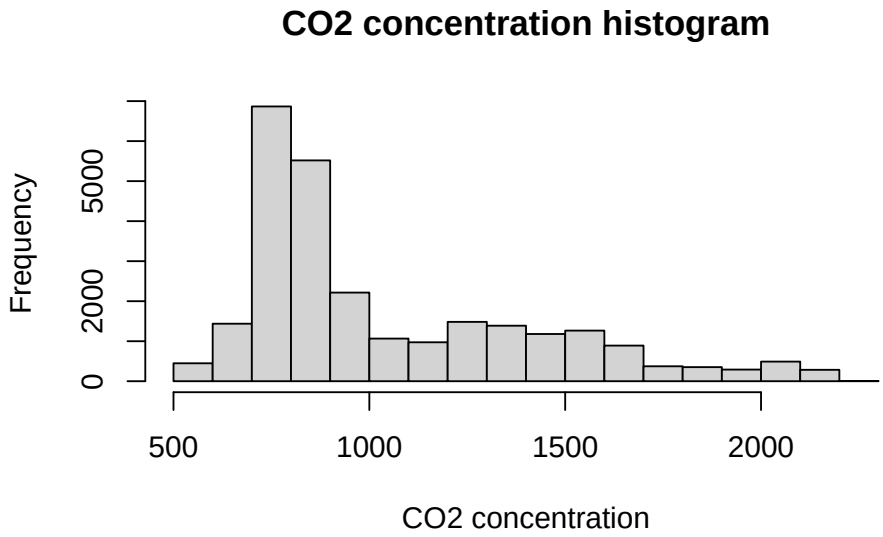
```
plot(co2$datetime, co2$co2, type="l", xlab="Date", ylab="CO2 concentration")
```



type="l" は線グラフを指定します。xlab と ylab で軸のラベルを指定します。

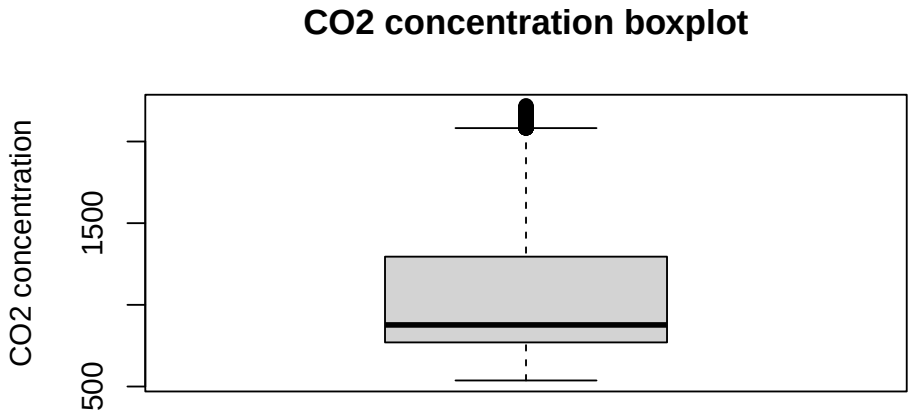
頻度分布を調べるために、ヒストグラムを描いてみましょう。

```
hist(co2$co2, breaks=20, main="CO2 concentration histogram", xlab="CO2 concentrat
```



箱ひげ図を描いてみましょう。

```
bp <- boxplot(co2$co2, main="CO2 concentration boxplot", ylab="CO2 concentration")
```



箱は四分位数を表し、箱の中の線は中央値を表します。ひげは四分位範囲の 1.5 倍の範囲を表しその範囲を超える値は外れ値と言います。点で示されている外れ値を取り除くには次のようにします。

```
co2 <- co2[co2$co2 >= bp$stats[1] & co2$co2 <= bp$stats[5], ]
```

3.2　AMeDAS

過去の AMeDAS データは、気象庁 > 各種データ・資料> 過去の気象データ・ダウンロードから取得できます。ファイル形式を参考にして読み解いていきます。地点や変数等を選択します。変数が多かったり、期間が長すぎたりすると、サイズの上限に達してしまいます。その場合は、必要な変数や期間を絞ってください。複数の地点を選択することもできますが、ここでは 1 地点を選んでください。ここでは、一部に欠測とみなすデータが含まれている、四日市における降水量の合計 (mm)、最高気温 (℃)、日照時間 (時間) について、2017 年 1 年分の日別値を取得しました。

読み込んだデータは R を使って整理することができます。例を見てみましょう。

列の名前は後でデータから拾ってつけるので、`header=FALSE` とします。第 1〜3 行目はダウンロードした時刻、空行、地点名などですので、`skip=3` で無視します。R 4.2 からは UTF-8 が標準になりました。AMeDAS データのファイル形式は Shift JIS（cp932）なので、`fileEncoding="cp932"` を渡します。

```
raw <- read.csv("data.csv", header=FALSE, skip=3, fileEncoding="cp932")
head(raw)
```

```
        V1                 V2                 V3                 V4                 V5
1     年月日 降水量の合計（mm） 降水量の合計（mm） 降水量の合計（mm） 降水量の合計（mm）
2
3                                       現象なし情報           品質情報             均
質番号
4 2017/1/1                  0                  1                  8                  1
5 2017/1/2                  0                  1                  8                  1
6 2017/1/3                0.0                  0                  8                  1
            V6             V7             V8             V9             V10
1 最高気温（℃） 最高気温（℃） 最高気温（℃） 日照時間（時間） 日照時間（時間）
2
3                    品質情報       均質番号                   現象なし情報
4         12.2             8              1            3.5                  0
5         12.9             8              1            8.2                  0
6         12.8             8              1            5.7                  0
              V11              V12
1 日照時間（時間） 日照時間（時間）
2
3          品質情報         均質番号
```

```
4               8               1
5               8               1
6               8               1
```

2 行目の NA を消しておきます。raw[3,] == "品質情報" は 2 行目の各要素が「品質情報」であれば TRUE、それ以外は FALSE であるベクトルです。

```
raw[3, is.na(raw[3,])] <- ""
raw[3,] == "品質情報"
```

```
     V1    V2    V3   V4    V5    V6   V7    V8    V9   V10  V11   V12
3 FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE
```

これを列の論理添字として TRUE の列について 7 より大きいか調べ、論理ベクトルを作ります。出力が長いので

```
head(raw[,raw[3,] == "品質情報"] > 7)
```

```
        V4    V7   V11
[1,]  TRUE  TRUE  TRUE
[2,] FALSE FALSE FALSE
[3,]  TRUE  TRUE  TRUE
[4,]  TRUE  TRUE  TRUE
[5,]  TRUE  TRUE  TRUE
[6,]  TRUE  TRUE  TRUE
```

apply() は R の強力な函数で、1 番目の引数として渡す配列の次元（MARGIN、2 番目の引数）に対して、3 番目に渡す函数を適用します。ここでは all を渡します。

```
head(apply(raw[,raw[3,] == "品質情報"] > 7, 1, all))
```

```
[1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
```

均質番号と品質情報以外のデータの列を特定します。

```
!(raw[3, ] == "均質番号" | raw[3,] == "品質情報")
```

```
    V1   V2   V3    V4    V5   V6    V7    V8   V9  V10   V11   V12
3 TRUE TRUE TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE FALSE FALSE
```

品質情報が全て 8（正常値）である行、データが含まれる列を残します。

```
filtered <- raw[apply(raw[,raw[3,] == "品質情報"] > 7, 1, all),
                !(raw[3, ] == "均質番号" | raw[3,] == "品質情報")]
head(filtered)
```

```
        V1                 V2                 V3          V6                 V9
1    年月日 降水量の合計（mm） 降水量の合計（mm） 最高気温（℃） 日照時間（時間）
3                                    現象なし情報
4 2017/1/1                  0                  1        12.2                3.5
5 2017/1/2                  0                  1        12.9                8.2
6 2017/1/3                0.0                  0        12.8                5.7
7 2017/1/4                0.0                  0        12.9                3.4
                V10
1 日照時間（時間）
3    現象なし情報
4                0
5                0
6                0
7                0
```

1 行目と 2 行目をつなげて列の名前として使います。

```
names(filtered) <- paste(filtered[1,], filtered[2,])
names(filtered)
```

```
[1] "年月日 "                              "降水量の合計（mm） "
[3] "降水量の合計（mm） 現象なし情報" "最高気温（℃） "
[5] "日照時間（時間） "                    "日照時間（時間） 現象なし情報"
```

R で負の添え字は、その添え字を取り除くことを意味します。Numpy のように後ろから数えるのではないことに注意しましょう。1〜2 行目はデータではないので削除します。

```
filtered <- filtered[-(1:2),]
head(filtered)
```

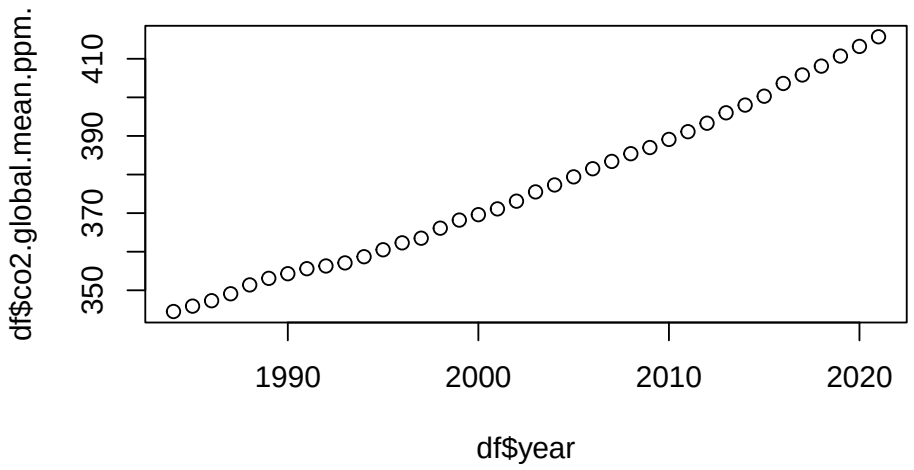
```
    年月日 降水量の合計（mm） 降水量の合計（mm） 現象なし情報 最高気温（℃）
4 2017/1/1                  0                                1        12.2
5 2017/1/2                  0                                1        12.9
6 2017/1/3                0.0                                0        12.8
7 2017/1/4                0.0                                0        12.9
8 2017/1/5                  0                                1        10.0
9 2017/1/6                  0                                1         9.5
  日照時間（時間） 日照時間（時間） 現象なし情報
```

4	3.5	0
5	8.2	0
6	5.7	0
7	3.4	0
8	7.7	0
9	7.6	0

3.3 大気微量成分

大気の中には主要成分だけでなく、量は少ないが重要な働きをする微量成分があります。二酸化炭素は人為起源の排出が原因で増加し続けています。その様子をグラフにしてみましょう。気象庁の Word Data Centre for Greenhouse Gases https://gaw.kishou.go.jp/publications/global_mean_mole_fractions から年平均 csv データ（Global annual mean mole fractions）の CO2 ファイルをダウンロードし、作業ディレクトに保存します。

```
df <- read.csv("co2_annual_20221026.csv")
plot(df$year, df$co2.global.mean.ppm.)
```



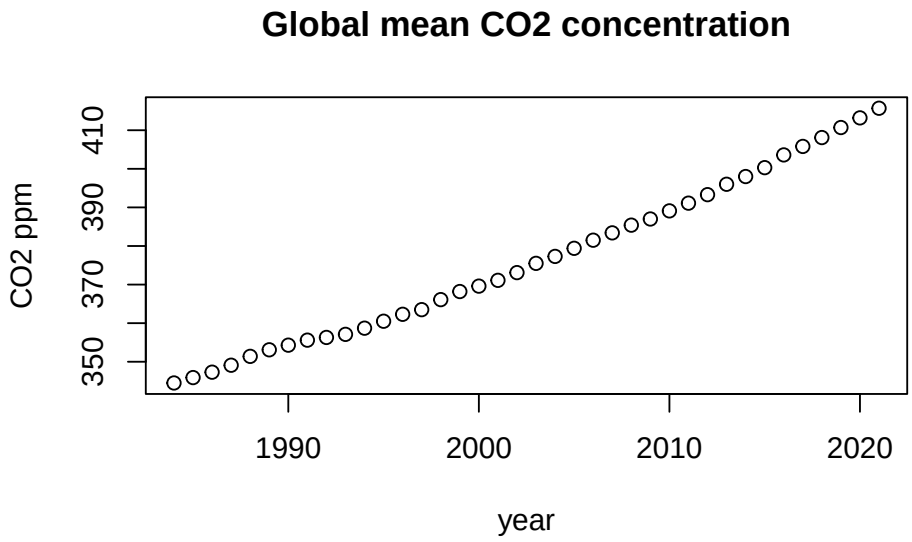
`read.csv()` はコンマ区切り（CSV, comma separated value）のデータを読む函数です。ファイル名は文字列であることを表すために `""` で囲みます。ファイル名の日付の部分は変わります。グラフの種類を指定しないと、散布図になります。

`df` には読んだ表の中身が入ります。`df` はデータフレームと呼ばれるクラスのデータ構造です。データフレームは、値が行列に並んでいるだけでなく、行や列に名前がつけられます。

グラフにタイトルをつけ、軸のラベルを変更します。タイトルは `main` で、軸ラベルは

xlab、ylab で指定します。

```
plot(df$year, df$co2.global.mean.ppm.,
     main="Global mean CO2 concentration",
     xlab="year", ylab="CO2 ppm")
```



次に月平均データ（Global monthly mean mole fractions）を使って簡単な時系列解析をしてみます。

```
dfm <- read.csv("co2_monthly_20231115.csv")
```

月毎の平均を求めてみます。

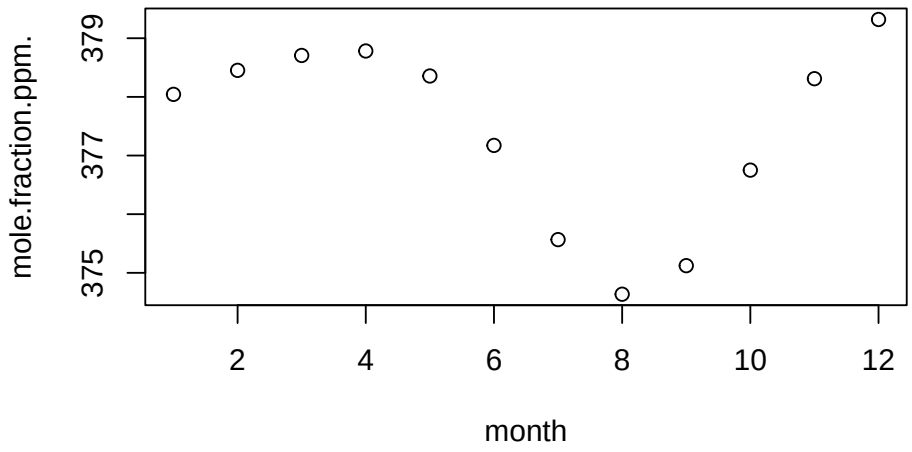
```
co2.annual.cycle <- aggregate(mole.fraction.ppm. ~ month, data = dfm, mean)
co2.annual.cycle
```

```
   month mole.fraction.ppm.
1      1           378.0431
2      2           378.4523
3      3           378.7062
4      4           378.7826
5      5           378.3562
6      6           377.1728
7      7           375.5664
8      8           374.6354
9      9           375.1223
10    10           376.7510
```

```
11     11            378.3100
12     12            379.3197
```

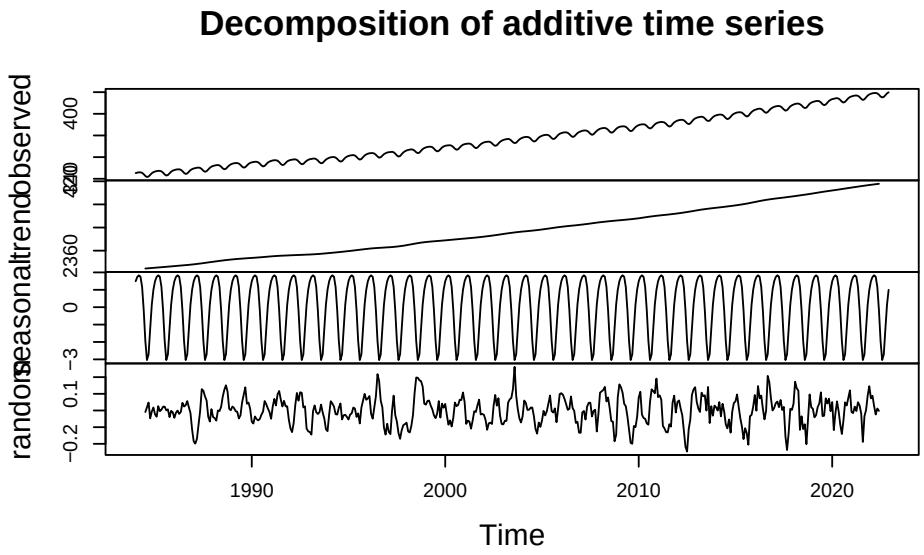
プロットしてみましょう。

```
plot(co2.annual.cycle)
```



8 月に一番少なく、1 月に一番多くなっています。

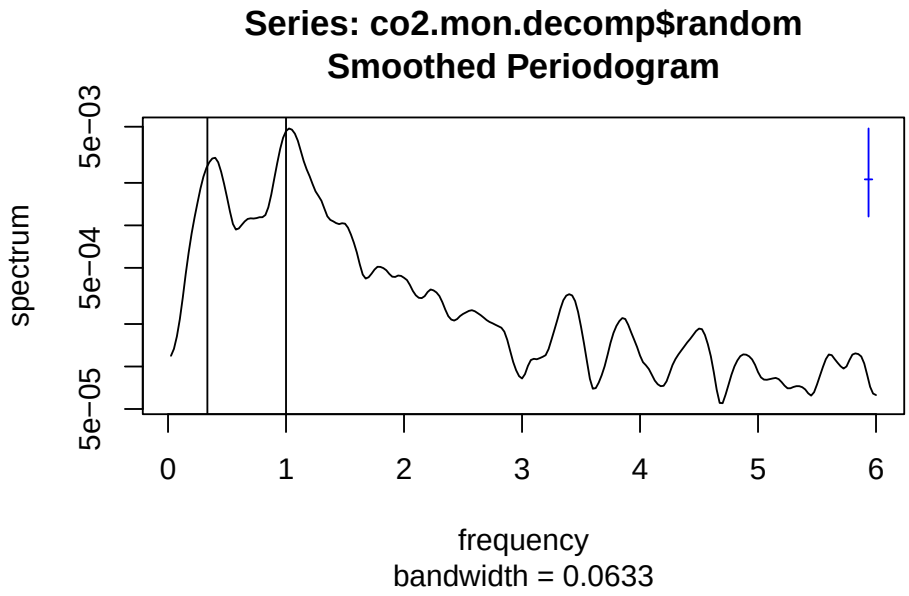
```
co2.mon <- dfm$mole.fraction.ppm.
n <- nrow(dfm)
co2.mon <- ts(co2.mon, start=c(dfm$year[1], dfm$month[1]), frequency=12)
co2.mon.decomp<- decompose(co2.mon)
plot(co2.mon.decomp)
```



`ts()` は一つ目の引数のデータから時系列オブジェクトを作ります。`frequency=12` は 1 年を単位として 12 回の頻度であることを示します。`decompose()` はデータをトレンド、周期成分と残差に分解します。

残差をさらにスペクトル分解してみましょう。

```
spec.pgram(co2.mon.decomp$random, spans=c(7,7), na.action=na.omit)
abline(v=1)
abline(v=1/3)
```



`spec.pgram` は高速フーリエ変換を利用して、ピリオドグラムを計算します。`spans` で修正ダニエル法に基づいた平滑化を施すことができます。右上の青い線は信頼区間を表しま

す。co2.mon.decomp$random には最初と最後に数値でない値（not a number）を na が入っていますので、na.action で無視します。ピークに近い、周期 1 年と 3 年に対応するところに縦線を入れてみました。それぞれ年々変動とエルニーニョ現象のような数年周期に対応しているものと考えられます。

R では、このように対話的な簡単な操作により、簡単に解析やグラフの作成ができます。

練習

自分で計測した CO2 データをグラフにしてみましょう。
df というテーブルがあり、date という名前の列に 2024/7/15、time という列に 12:00:00 などのようにデータが入っているとします。これらを R の日時を表す型に変換して、列として追加するには次のようにします。

```
df$datetime <- as.POSIXct(paste(df$date, df$time))
```

POSIXct 型は 1970 年 1 月 1 日 00:00:00 からの秒数です。これを横軸に取ると良いでしょう。
気象庁は様々なデータを CSV 形式で提供しています。それらを R で読んでみましょう。うまく読めるでしょうか。

- 「気象業務はいま 2021」CSV データ一覧
- 台風位置表
- 数値データページリンク集
- 最新の気象データ

第 4 章

回帰分析

4.1 統計

R では統計量を簡単に計算できます。ベクトルに対する函数をいくつか挙げます。

統計量	R の函数	統計量	R の函数
平均	`mean()`	中央値	`median()`
最大値	`max()`	最小値	`min()`
範囲	`range()`	累積和	`cumsum()`
和	`sum()`	積	`prod()`
逆順	`rev()`	整列化	`sort()`
順位	`rank()`	元位置	`order()`

- `sort()` は小さい順に整列化します。
- `rank()` は整列化前の元の並びでの整列化の順位を返します。
- `order()` は整列化後に各要素が整列化前にいた位置を返します。

4.2 線型回帰

回帰分析（regression analysis）は、二つの変数の x と y のデータが与えられたときに、$y = f(x)$ というモデルを当てはめます。x を説明変数、y を目的変数と呼び、x がスカラーの場合を単回帰 (simple regression)、ベクトルの場合を重回帰 (multiple regression) と言います。モデルが線型 $f(x) = \beta_0 + \beta_1 x$ なら、線型回帰、$f(x) = ax^b, f(x) = ae^{bx}, f(x) = a + b\log x$ のように非線型なら非線型回帰といいます。様々な当てはめ（curve-fitting）があります。

n 個の観測 $y_1, \ldots, y_n$ が与えられたとき、線型モデルでは次のように書けます。

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{4.1}$$

ε_i は誤差を表しています。最小分散推定（minimum variance）は最小二乗法（method of least squares）とも呼ばれ、誤差の分散

$$S = \sum_i \varepsilon_i^2 \tag{4.2}$$

を最小にする β_0, β_1 を求めます。

Equation 4.2 を β_0, β_1 で微分して 0 と置くと、正規方程式

$$\begin{aligned} \frac{\partial S}{\partial \beta_0} &= -2\sum[y_i - (\beta_0 + \beta_1 x_i)] = 0 \\ \frac{\partial S}{\partial \beta_1} &= -2\sum[y_i - (\beta_0 + \beta_1 x_i)]x_i = 0 \end{aligned} \tag{4.3}$$

平均

$$\bar{x} = \frac{1}{n}\sum x_i,\ \bar{y} = \frac{1}{n}\sum y_i$$

を用いると、Equation 4.3 の最初の式から

$$\bar{y} = \beta_0 + \beta_1 \bar{x} \tag{4.4}$$

が得られます。

$$\begin{aligned} \sum(x_i - \bar{x})\bar{x} &= \bar{x}\sum x_i - n\bar{x}^2 = 0 \\ \sum(y_i - \bar{y})\bar{x} &= \bar{x}\sum y_i - n\bar{x}\bar{y} = 0 \end{aligned}$$

を用いると

$$\begin{aligned} \hat{\beta}_0 &= \bar{y} - \beta_1 \bar{x} \\ \hat{\beta}_1 &= \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{\mathrm{cov}(x, y)}{\mathrm{var}(x)} \end{aligned}$$

を得ます。$\hat{\beta}_1$ を回帰係数といいます。

```
df <- read.csv("co2_annual_20221026.csv")
beta1 <- cov(df$year, df$co2.global.mean.ppm.) / var(df$year)
beta0 <- mean(df$co2.global.mean.ppm.) - beta1 * mean(df$year)
print(c(beta0, beta1))
```

```
[1] -3427.371014     1.899497
```

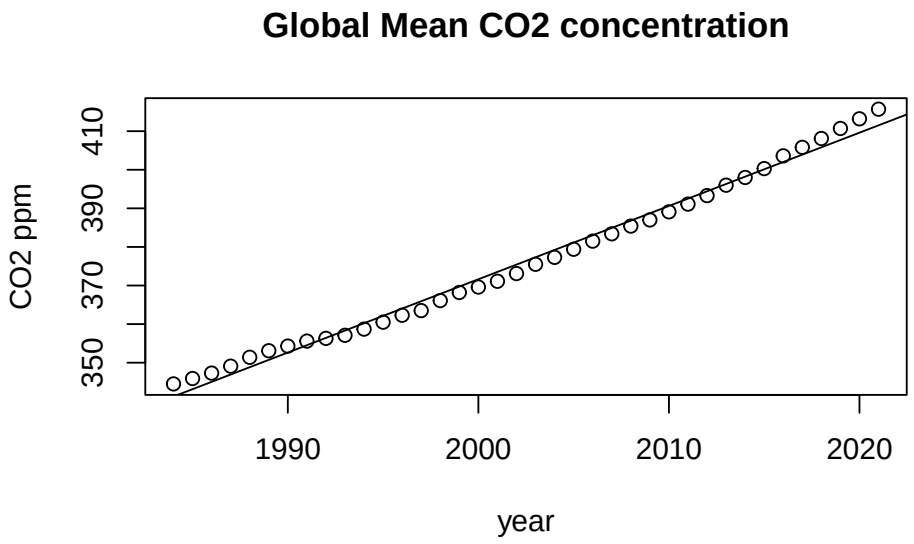
実は R には回帰分析をする函数 `lm()` があります。

```
lm.co2 <- lm(df$co2.global.mean.ppm. ~ df$year)
print(lm.co2$coefficients)
```

```
 (Intercept)       df$year
-3427.371014     1.899497
```

グラフに回帰直線を入れてみましょう。

```
plot(df$year, df$co2.global.mean.ppm.,
     main="Global Mean CO2 concentration",
     xlab="year", ylab="CO2 ppm")
abline(lm.co2)
```



4.3 非線型回帰

いつも線型回帰が当てはまるとは限りません。(Anscombe 1973) の例を描画してみます。`?anscombe` に掲載されています。

```
ff <- y ~ x                                                          ①
par(mfrow = c(2, 2), mar = 0.1+c(4,4,1,1), oma =  c(0, 0, 2, 0))     ②
for (i in 1:4){
  ff[2:3] <- lapply(paste0(c("y", "x"), i), as.name)                 ③
  plot(ff, data=anscombe, col="red", pch=21, bg="orange",
       xlim=c(3, 19), ylim=c(3, 13))                                 ④
  abline(lm(ff, data=anscombe), col="blue")
}
```

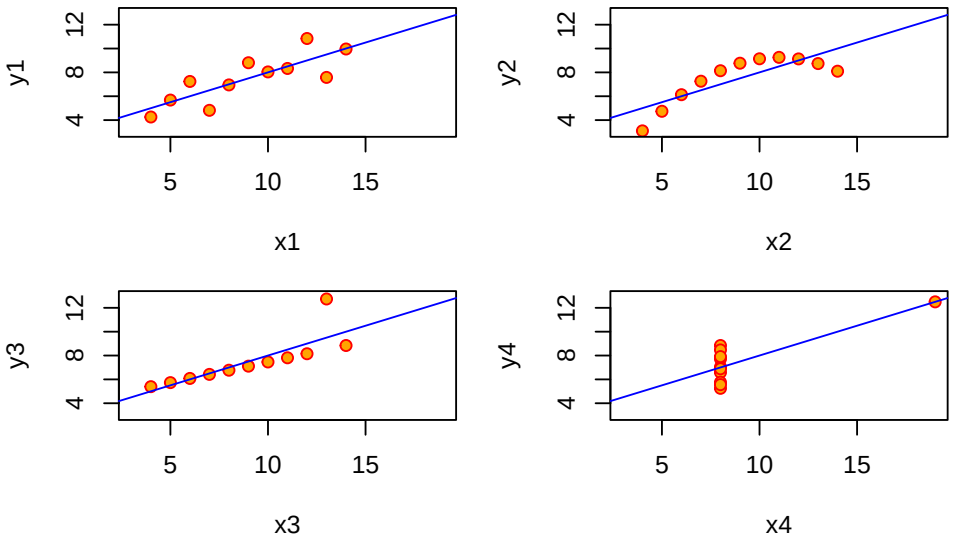
① `ff` に線型モデルを代入します。

② `par` でパラメタを設定します。`mfrow` で 2×2 枚のパネルを作ります。余白（`mar`）

と外部余白（oma）を設定します。下から時計回りに左、上、右の順です。

③ lapply はリストやベクトルに函数を適用します。paste0 で区切り文字なしで文字 y または x とループの番号 i を連結します。この文字列ベクトルに as.name を適用して R のオブジェクト名にします。ff の 1 番目の要素には函数 ~() が入っているので、2 番目と 3 番目の要素として引数を与えます。最初のループは i が 1 なので y1 と x2 です。

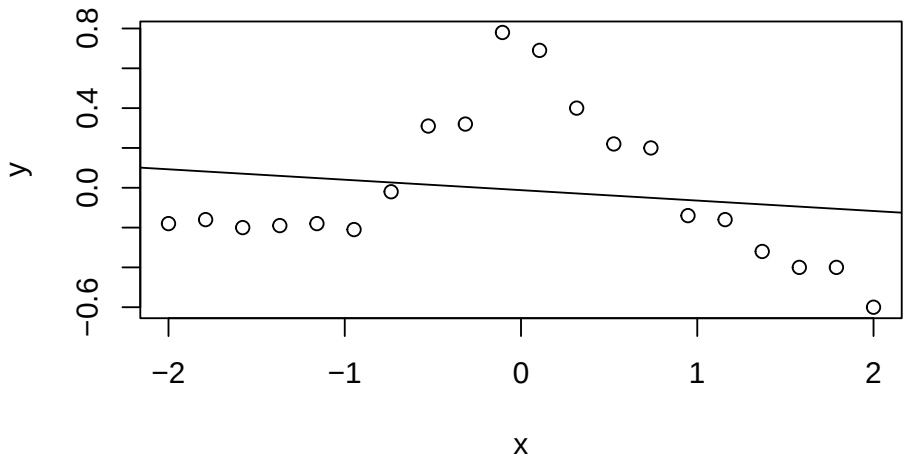
④ y ~ x を最初の引数として散布図を描きます。x, y とは順序が入れ替わっているので注意が必要です。データフレームを data= で指定します。



anscomb の 4 つのデータは x の平均と分散は完全に一致し、y の平均と分散もほぼ等しいので、相関係数や回帰直線もほとんど同じです。左上は線型で良さそうです。右上は線型ではなく二次曲線の一部のように見えます。左下は外れ値の影響を受けています。右下もたった一つの外れ値のために、線型ではないのに相関係数が高くなってしまっています。統計量を計算するだけではなく、描画することが重要です。

次に 赤穂昭太郎 (2008) の例を示します。値は目分量なので、結果は教科書と同じではありません。

```
n <- 20
x <- seq(-2, 2, length.out=n)
y <- c(-0.18, -0.16, -0.2, -0.19, -0.18, -0.21, -0.02,
       0.31, 0.32, 0.78, 0.69, 0.4, 0.22, 0.2, -0.14,
       -0.16, -0.32, -0.4, -0.4, -0.6)
plot(x,y)
abline(lm(y ~ x))
```



ガウス函数の線型結合

$$f(x) = \sum_{j=1}^{k} \alpha_j \exp(-\beta(x_j - x)^2)$$

を用いると複雑な函数を表現できます。$y - f(x)$ に対して最小二乗法を適用します。ただし過学習を防ぐために正則化も併用します。ガウス函数を要素とする $k \times k$ の行列を $\mathbf{K}$ で表します。

$$J(\alpha) = (\mathbf{y} - \mathbf{K}\alpha)^{\mathrm{T}}(\mathbf{y} - \mathbf{K}\alpha) + \lambda\alpha^{\mathrm{T}}\mathbf{K}\alpha$$

を最小化します。$\mathbf{K}$ が正則であることを利用すると、重みは

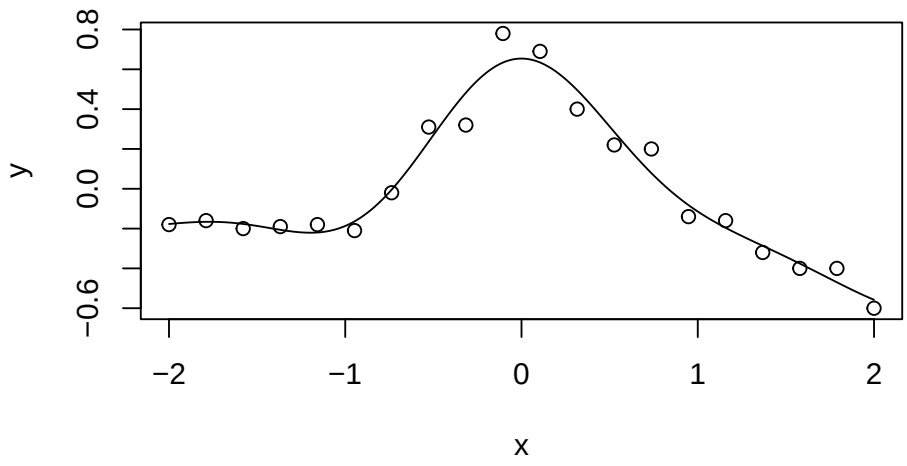
$$\alpha = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

と定ります。このような手法をカーネル回帰と呼び、ここではガウス函数をカーネルとして用いました。$\mathbf{K}$ は Gram 行列と呼びます。

$\beta = 1, \lambda = 0.01$ の場合は次のようになります。

```
lambda <- 0.01
calc.ga <- function(x, y, beta=1.0) {
  exp(-beta * (x - y)^2)
}
x.10 <- seq(-2, 2, length.out=n*10)
kmat <- outer(x, x, calc.ga)
alpha <- solve((kmat + lambda * diag(n)), y)
```

```
y.10 <- outer(x.10, x, calc.ga) %*% alpha
plot(x, y)
lines(x.10, y.10)
```



4.4　交差検証

回帰で推定された函数が一般性を持つかどうかの汎化能力を評価するため、交差検証を行なってみましょう。標本から一部を除いて学習し、除いたデータで評価を行います。これを繰り返して平均した誤差が交差検証誤差です。線型回帰やカーネル回帰の場合、データを一つずつ除く交差検証は次の式で評価できます。

$$\mathrm{CV} = \frac{1}{n}\sum_{i=1}^{k}\left(\frac{y_i - \hat{y}_i}{1 - H_{ii}}\right)^2$$

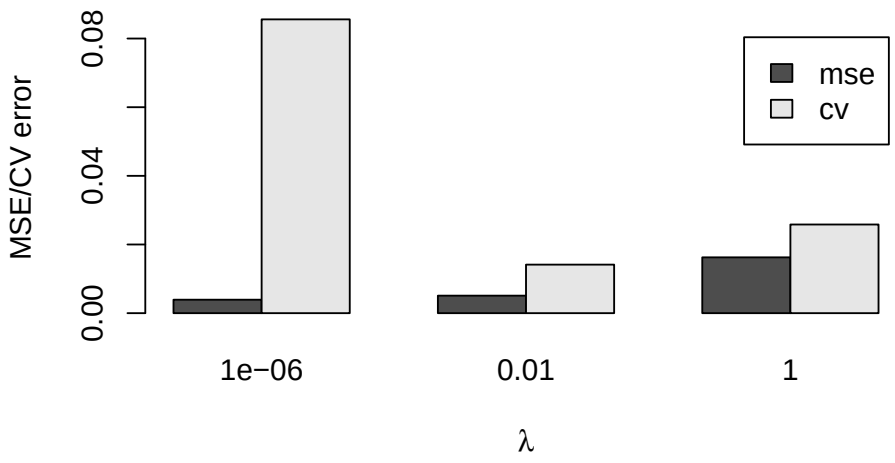
ここで回帰による推定値は $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$ で求めます。カーネル回帰の場合 $\mathbf{H} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{K}$ です。

$\beta = 1$ に固定して λ を変えた場合の平均二乗誤差と交差検証誤差を求めてみましょう。

```
calc.mse <- function(y, yf) {
  mean((y - yf)^2)
}
calc.cv <- function(y, yf, hii){
  mean(((y - yf) / (1 - hii))^2)
}
lambda=c(1.0e-6, 0.01, 1.0)
```

```
mse=rep(0, length(lambda))
cv=rep(0, length(lambda))

i <- 1
for (l in lambda) {
  hmat <- solve((kmat + l * diag(n)), kmat)
  hii <- diag(hmat)
  alpha <- solve((kmat + l * diag(n)), y)
  yf <- outer(x, x, calc.ga) %*% alpha
  mse[i] <- calc.mse(y, yf)
  cv[i] <- calc.cv(y, yf, hii)
  i <- i + 1
}
error <- rbind(mse, cv)
rownames(error) <- c("mse", "cv")
colnames(error) <- lambda
barplot(error, beside=TRUE, legend=TRUE,
        xlab=expression(lambda), ylab="MSE/CV error")
```



パラメタは二つの誤差が共に小さくなるように決めます。

練習

- β や λ を変えてカーネル回帰をしてみましょう。
- λ を固定して、β を変えたとき、二つの誤差を計算して教科書と比較してみましょう。

第 5 章

主成分分析

主成分分析は，回帰とともによく用いられる統計手法です。主成分分析は，ビッグデータを要約することができます。気象学では経験的直交函数（Empirical Orthogonal Functions, EOF），機械学習では次元削減手法として用いられる特異値分解，様々な分野の数値解析に用いられる固有値解析としても知られています。R を使って手を動かしながらどのような手法か学んでいきましょう。

5.1 特異値解析

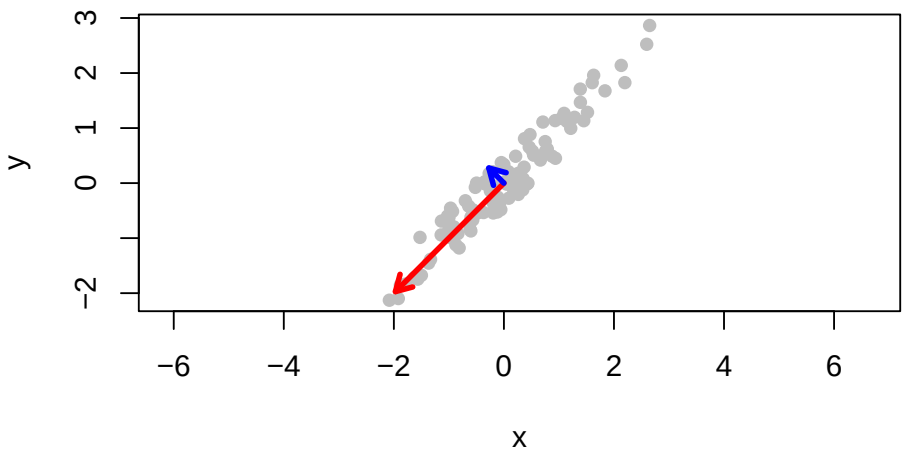
100 個の観測が 2 組得られたとします。二つの組には何らかの関係があります。ばらつきの大きい方向を特異値解析で求めましょう。

```
m <- 100
set.seed(514)

x <- rnorm(m)
x <- (x - mean(x)) / sd(x)
y <- x + runif(m)
y <- (y - mean(y)) / sd(y)

amat <- cbind(x, y)
usv <- svd(amat)
v1 <- usv$d[1] * usv$v[, 1] * 0.2
v2 <- usv$d[2] * usv$v[, 2] * 0.2

plot(x, y, col="gray", pch=16, asp=1)
arrows(0, 0, v1[1], v1[2], length=0.1, lwd=3, col="red")
arrows(0, 0, v2[1], v2[2], length=0.1, lwd=3, col="blue")
```



乱数で x を生成し，それを少し乱した y を作ります。標準化した後 100×2 の配列 amat にまとめます。これを svd() で特異値分解をしてばらつきの大きい方向（固有ベクトル）を求めています。●はデータ，→は右特異ベクトルで特異値に比例させています。赤，青はそれぞれ 1 番目，2 番目に分散が大きい方向を表します。

カラーパレット

カラーパレットは palette() にパレット名を渡すと確認できます。引数なしだと RGB の HEX 値が表示されます。パレットの名前は palette.pals() で確認できます。

```
palette.pals()
```

```
 [1] "R3"              "R4"              "ggplot2"         "Okabe-
Ito"
 [5] "Accent"          "Dark 2"          "Paired"          "Pastel 1"
 [9] "Pastel 2"        "Set 1"           "Set 2"           "Set 3"
[13] "Tableau 10"      "Classic Tableau" "Polychrome 36"   "Alphabet"
```

既定のカレーパレットは R4 です。

```
k <- length(palette("R4"))
par(mar=rep(0, 4)); plot.new(); plot.window(c(0, k+1), c(0, 0.15))
points(1:k, rep(0.1, k), col=1:k, pch=15, cex=2)
text(1:k, 0.1, pos=rep(c(1, 3), length.out=k), palette(), col=1:k)
```

matplotlib で採用されている、tableau 風の配色は Tableau10 です。

```
k <- length(palette("Tableau10"))
par(mar=rep(0, 4)); plot.new(); plot.window(c(0, k+1), c(0, 0.15))
points(1:k, rep(0.1, k), col=1:k, pch=15, cex=2)
text(1:k, 0.1, pos=rep(c(1, 3), length.out=k), palette(), col=1:k)
```

5.2 特異値解析と固有値解析との関係

ここで，特異値解析と固有値解析との関係についておさらいをしておきます。ここではデータは実数とします。特異値分解は，m 個のデータ n 組を並べた $m \times n$ 行列 $\mathbf{X}$ を

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{5.1}$$

のように分解するものです。$\mathbf{U}$ は $m \times m$ 行列，$\mathbf{S}$ は $m \times n$ 行列，$\mathbf{V}$ は $n \times n$ 行列です。$\mathbf{U}$ は左特異ベクトル，$\mathbf{V}$ は右特異ベクトルと呼ばれています。特異値は $\mathbf{S}$ の対角成分として最大 m と n の小さい方の個数（$p = \min(m, n)$）が得られます。$\mathbf{U}$ と $\mathbf{V}$ はともに直交行列で $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}$，$\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$ が成り立ちます。

```
print(usv$d)
```

```
[1] 13.937893  1.932649
```

```
print(usv$v)
```

```
           [,1]       [,2]
[1,] -0.7071068 -0.7071068
[2,] -0.7071068  0.7071068
```

```
wv <- eigen(t(amat) %*% amat)
print(sqrt(wv$values))
```

```
[1] 13.937893  1.932649
```

```
print(wv$vectors)
```

```
          [,1]       [,2]
[1,] 0.7071068 -0.7071068
[2,] 0.7071068  0.7071068
```

第 6 章

函数

ジオポテンシャル高度の定義 Equation 2.1 では、重力加速度は高さの函数としました。重力加速度は高さによりどのように変化するのでしょうか。重力加速度は万有引力の法則に従い、地球の中心からの距離 $a + z$（a は地球半径）の 2 乗に反比例します。万有引力定数を G、地球の質量を M とすると重力加速度 g は地表面からの幾何学的な高さ z の函数として、次の式で表されます。

$$g(z) = \frac{GM}{(a+z)^2} \tag{6.1}$$

$G = 6.6743 \times 10^{-11} \mathrm{m}^3\,\mathrm{kg}^{-1}\,\mathrm{s}^{-2}$、地球の質量を $M = 5.9742 \times 10^{24} \mathrm{kg}$ を使います。

6.1 重力加速度

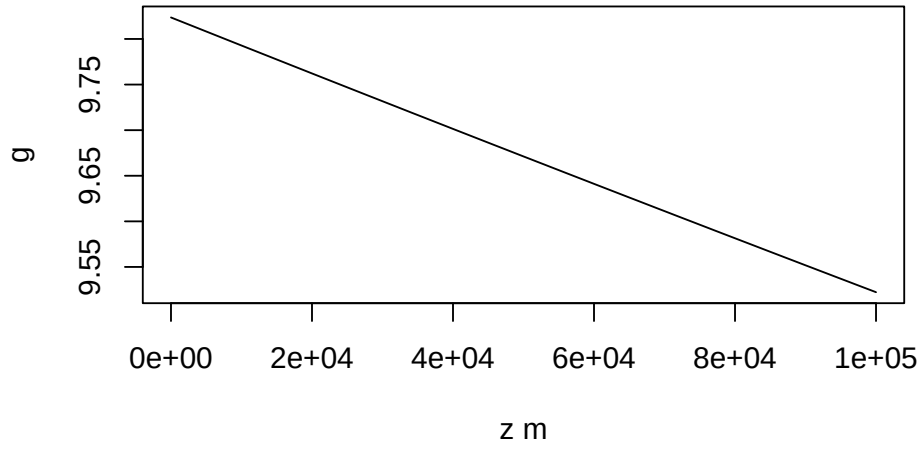
様々な z について計算できるように、函数を作ります。R の函数は `function` を使って定義します。( ) の中の `z` は引数と呼ばれています。R では最後に評価された値が戻ります。

```
G <- 6.6743e-11
M <- 5.9742e24
a <- 6.371e6
calc.g <- function(z) {
  G * M / (a + z)^2
}
```

`calc.z(0)` とコンソールに入れれば地表面での重力加速度が計算できます。`g0 <- calc.z(0)` とすると値が `g0` に入ります。10 km では、100 km ではどんな値になるでしょうか。

図に描いてみましょう。

```
plot(calc.g, 0, 1e5, xlab="z m", ylab="g")
```



何度もコマンドを入力しなくてもいいように、以下のようなスクリプトファイルにまとめて保存しましょう。

束にまとめて一括して処理するという意味で、スクリプトを使う方法をバッチ処理と呼びます。これに対して、コンソールにコマンドをひとつひとつ入れる対話（インタラクティブ）処理と呼びます。対話処理によりデータについて調べる探索的データ分析は R の特徴ですが、その場合にも函数を書いておくと効率がよくなります。

上のコードを `gravity.R` というファイルに保存してください。source ボタンをクリックするとスクリプトが実行され、定義した函数がコンソールで使えるようになります。

下の 3 行は `plotGravity.R` というファイルに保存しましょう。

```
pdf("gravity.pdf")
plot(calc.g, 0, 1e5, xlab="z m", ylab="g")
dev.off()
```

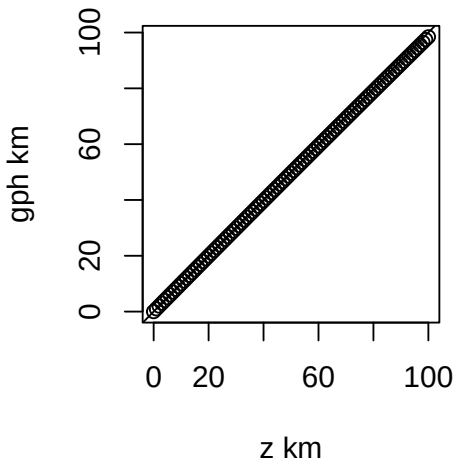
`plot()` の前後に追加された 2 行について説明します。`pdf()` でファイル名を指定しています。描画コマンドは `dev.off()` より前に描いてください。出力ファイル名を指定しない場合は `Rplots.pdf` に出力されます。PNG 形式の場合は代わりに `png()` を使います。

スクリプトは Run ボタンを押すと実行されます。

6.2 幾何高度とジオポテンシャル高度

```
calc.gph <- function(z) {
  integrate(calc.g, 0, z)$value / calc.g(0)                    ①
}
z <- 0:100                                                     ②
gph <- sapply(z * 1000, calc.gph) / 1000                       ③
par(pty="s")                                                   ④
plot(z, gph, xlab="z km", ylab="gph km")
abline(0, 1)                                                   ⑤
```

① `integrate()` 函数を使って 0 から z まで数値積分します。
② 0 から 100 まで 1 刻みのベクトルを作ります。
③ `integrate()` の積分の上限はスカラーなので `sapply()` で `calc.gph()` を z の各要素に作用させます。
④ 描画領域を正方形（square）にします。
⑤ 切片 0、傾き 1 の直線を引きます。



練習

1. 上記のコードをスクリプトにしてみよう。
2. AMeDAS データに含まれる風向 16 方位を角度にする函数を書いてみよう。

```
dir2deg <- function(x) {
  dir <- seq(0, 360, length.out=17)[1:16]
  names(dir) <- c("北", "北北東", "北東", "東北東",
                  "東", "東南東", "南東", "南南東",
                  "南", "南南西", "南西", "西南西",
                  "西", "西北西", "北西", "北北西")
  dir[x]
}
dir <- c("北", "北北東", "北東", "東北東")
dir2deg(dir)
```

```
   北 北北東    北東 東北東
  0.0  22.5  45.0  67.5
```

6.3　湿潤大気に関する函数

Bolton (1980) や世界気象機関 WMO、気象庁などに基づいて、湿潤大気に関する函数を書いてみました。gist にも掲載しているほか、met に含まれています。

devtools をインストールして、

```
library(devtools)
install_github("tenomoto/mettools")
```

とするとインストールされます。

```
eps <- 0.622

e2q <- function (e, p) {
  eps*e/(p-(1.0-eps)*e)
}

q2e <- function(q, p) {
    p*q/(eps+(1.-eps)*q)
}

e2w <- function(e, p) {
    eps*e/(p-e)
}

calc.es <- function(T) {
```

```
# WMO, JMA
    exp(19.482-4303.4/(T-29.65))*100
}

calc.condtemp <- function (T, e){
#; Bolton (1980)
#; e(Pa)
    2840.0/(3.5*log(T)-log(e*0.01)-4.805)+55.0
}

ttd2q <- function(ttd, T, p) {
    e2q(calc.es(T-ttd), p)
}

rh2q <- function (rh, T, p) {
# T K, rh %
    e2q(calc.es(T)*0.01*rh, p)
}

q2ttd <- function(q, T, p) {
# WMO, JMA
  T-29.65-4303.4/(19.482-log(q2e(q,p)*0.01))
}

calc.theta <- function (T, w, p) {
# Bolton (1980)
# T(K), w(kg/kg), p(Pa)
    T*(100000.0/p)^(0.2854*(1.0-0.28*w))
}

calc.thetae <- function (T, e, p) {
# Bolton (1980)
# T(K), e(Pa), p(Pa)
    w <- e2w(e,p)
    TL <- calc.condtemp(T, e)
    calc.theta(T, w, p) * exp((3376.0/TL-2.54)*w*(1.0+0.81*w))
}

calc.thetaes <- function(T, p) {
# Bolton (1980)
# T(K), e=es(T)(Pa), p(Pa)
```

```
    es <- calc.es(T)
    w <- e2w(es,p)
    calc.theta(T, w, p) * exp((3376.0/T-2.54)*w*(1.0+0.81*w))
}
```

第 7 章

海面水温

気象庁の海面水温データ MGDSST を可視化してみます。MGDSST は NEAR-GOOS から提供されています。ここから 2022 年 6 月 19 日の全球日平均海面水温をダウンロードし作業ディレクトリに移します。

データの 1 行目は日付と行と列の数が書かれています。MGDSST では 1 地点の海面水温（℃）の 10 倍を 3 文字で表しています。0.125E, 89.875N から北西から南東に向かってデータが並んでいます。海氷は 888、陸は 999 で表されています。

```
fname <- "mgd_sst_glb_D20220618.txt.gz"
nlon <- 1440
nlat <- 720
dlon <- 360 / nlon
dlat <- 180 / nlat
df <- read.fwf(gzfile(fname), widths=rep(3, nlon),
               header=F, skip=1, nrow=nlat,
               na.strings=c("888", "999"))                    ①
```

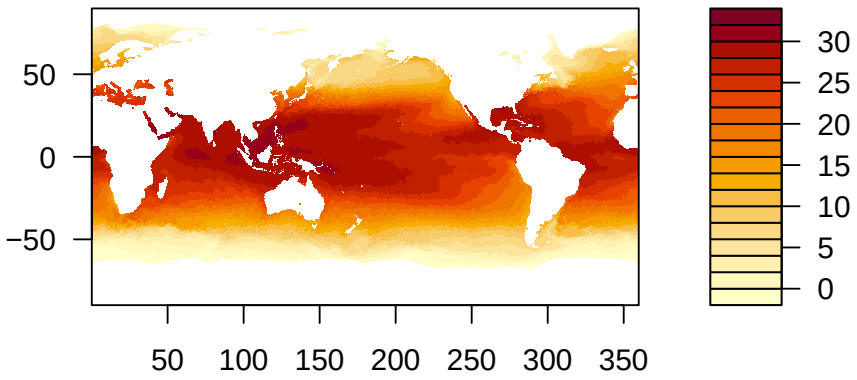
① 固定形式のテキストファイルを読む函数 `read.fwf()` を使って読みます。要素を繰り返す `rep()` を用いて 3 を 1440 個並べたベクトルを `widths` に与えます。ヘッダを使わないので `header=F` とします。`nrow=nlat` は行の数として緯度の数を与えています。海氷と陸は非数 `NA` とするため `na.strings=c("888", "999")` を渡します。

```
sst <- t(as.matrix(df)[nrow(df):1,]) * 0.1                    ①
lon <- seq(0+dlon/2, 360-dlon/2, dlon)                        ②
lat <- seq(-90+dlat/2, 90-dlat/2, dlat)                       ③
filled.contour(lon, lat, sst)                                 ④
```

① 読んだデータを行列に変換し、南から並べ替え転置して、0.1 倍します。
② 経度を定義します。

③　緯度を定義します。

④　filled.contour() で等値線を描きます。



日本域に限定して描いてみましょう。西と東の端の経度に対応するインデックスを探します。

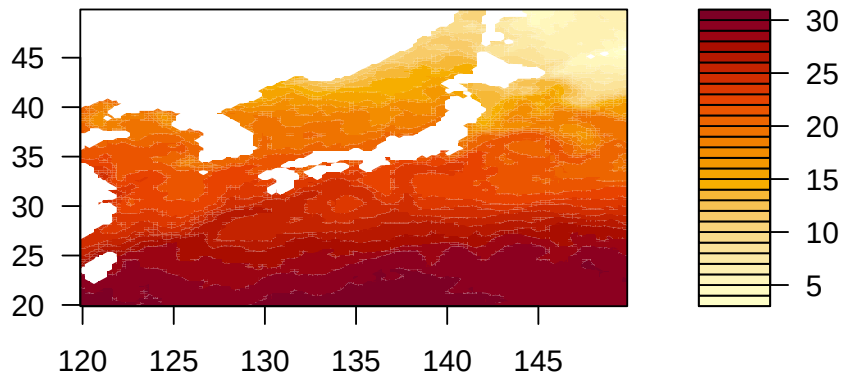
```
lon0 <- 120
lon1 <- 150
i0 <- which.min(abs(lon - lon0))
i1 <- which.min(abs(lon - lon1))
```

同様に南北の端も決めます。

```
lat0 <-  20
lat1 <-  50
j0 <- which.min(abs(lat - lat0))
j1 <- which.min(abs(lat - lat1))
```

求めたインデックスを使って経度と緯度、海面水温の範囲を指定して描きます。

```
filled.contour(lon[i0:i1], lat[j0:j1], sst[i0:i1,j0:j1])
```



練習

経度・緯度をインデックスにするコードを函数にしてみよう。好きな海域の海面水温を描いてみよう。

第 8 章

地図

NaturalEarth のデータを使って地図を描いてみます。

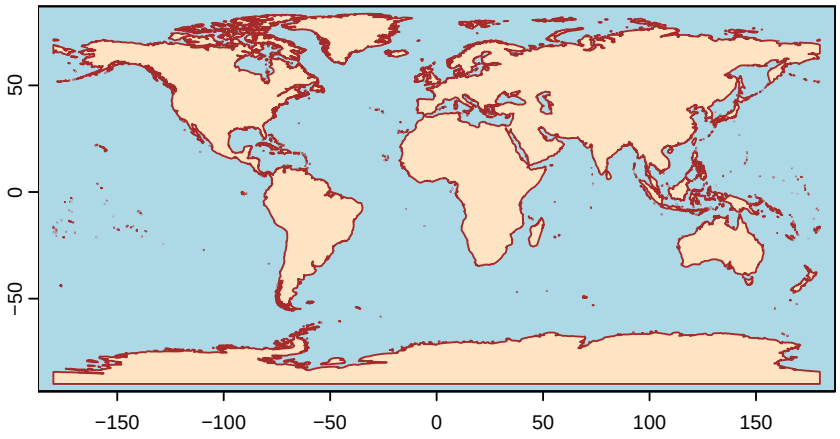
8.1 地理情報データ

点を結んだベクトルデータが shp 形式で、ラスタ（画像）データが tif 形式で提供されています。中解像度 5 千万分の 1（1:50m、1 cm = 500 km）自然（Physical）ベクトル形式の中から陸（Land）を使います。Downloads から目的のファイルへのリンクを見つけて、展開すると `ne_50m_land` というフォルダができます。フォルダごと作業ディレクトリに配置してください。

地理データを扱うパッケージ terra を使います。`vect()` に shp ファイルのパスを与えて読み込み、ベクトルオブジェクト（`SpatVector`）を生成します。

```
library(terra)

lshp <- "ne_50m_land/ne_50m_land.shp"
l50 <- vect(lshp)
plot(l50, border="brown", col="bisque", background="lightblue")
```

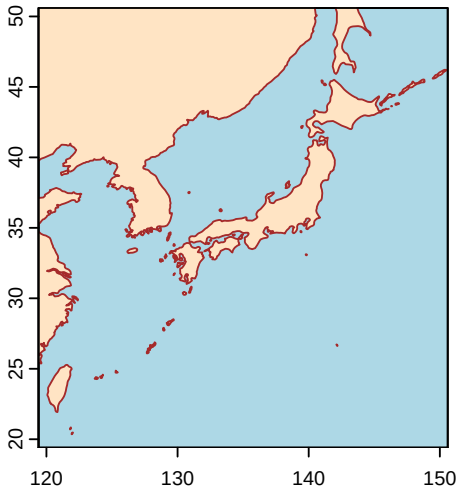


陸が多角形として定義されているので、col に薄い茶色ビスク "bisque" を指定し、背景 background となる海を水色 "lightblue" に塗り、境界 border である海岸線は茶色 "brown" にしました。R の色の名前は An overview of color names in R を参照してください。

8.2　描画範囲

描画範囲を日本域に限定してみましょう。

```
plot(l50, border="brown", col="bisque", background="lightblue",
     ext=ext(120, 150, 20, 50))
```

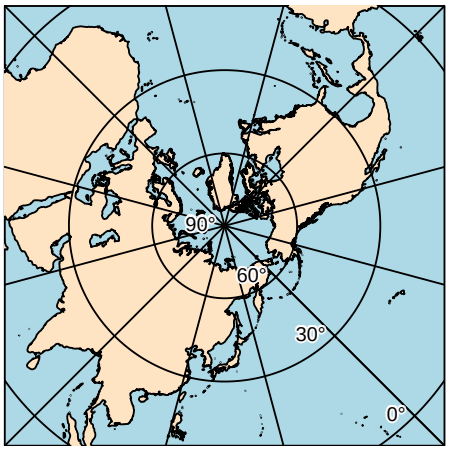


8.3 地図投影

地図投影には proj が使われています。投影したいベクトル座標参照系（crs: coordinate reference system）を指定する文字列を `project()` に与えます。投影については、Projections やや (Evenden 1991) を参照してください。

`axes=FALSE` で座標を消し、`ext` で描画範囲を絞ります。経線と緯線は `graticule` で引くことができます。

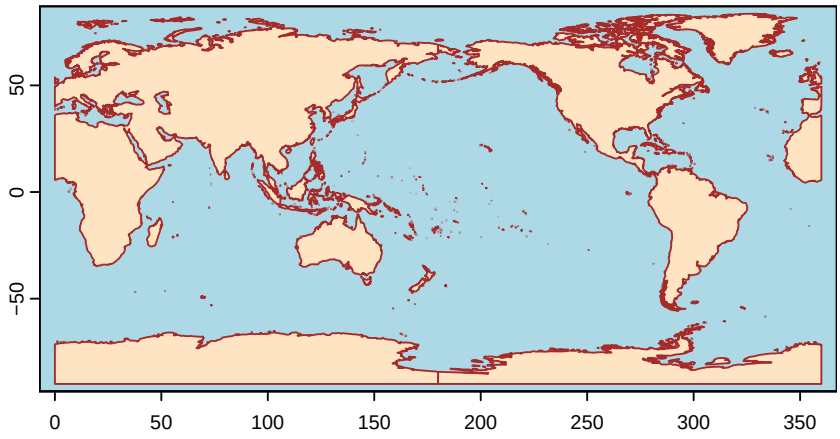
```
newcrs <- "+proj=stere +lon_0=135e +lat_0=90n"
l50p <- project(l50, newcrs)
plot(l50p, axes=FALSE, col="bisque", background="lightblue",
     ext=ext(-1e+7, 1e7, -1e7, 1e7))
g <- graticule(30, 30, crs=newcrs)
plot(g, add=TRUE)
```



8.4　ベクトルデータの処理

日付変更線を中心した地図を描いてみましょう。NaturalEarth のデータの経度は − 180° から 180° で、本初子午線 0° が中心です。西半球と東半球を切り取り、西半球の経度を正にずらして、貼り合わせれば良さそうです。1. `crop()` で東半球（0° から 180° まで）と西半球（ − 180° から 0° まで）をそれぞれ切り取ります。2. `shift()` で西半球を 360° ずらします。3. `rbind()` で二つをくっつけます。左端を本初子午線以外にすることができるように、函数にパラメタを与えられるようにし、既定値は 0 とします。

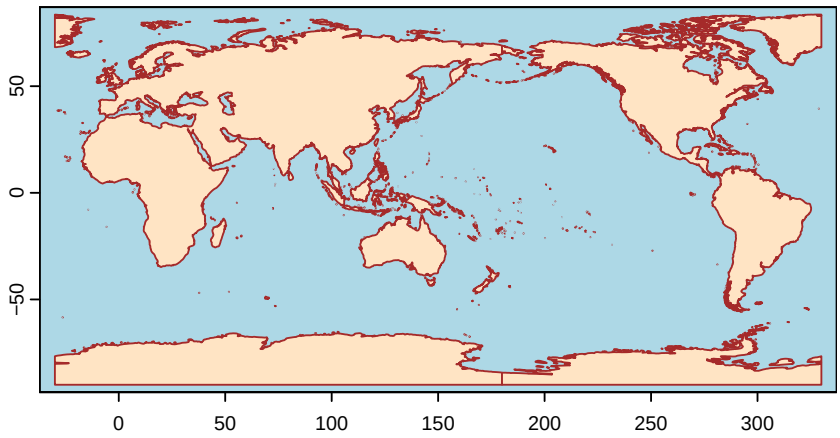
```
shift.lon <- function(v, dlon=0) {
  e <- crop(v, ext(dlon, 180, -90, 90))
  w <- crop(v, ext(-180, dlon, -90, 90))
  w <- shift(w, 360)
  rbind(e, w)
}
l50s <- shift.lon(l50)
plot(l50s, border="brown", col="bisque", background="lightblue")
```



よく見るとロシアのチュコト半島と南極に線が入っていますが、日付変更線を重ねると隠れるので気にしないことにします。

30°W からの地図も描いてみます。

```
l50s <- shift.lon(l50, -30)
plot(l50s, border="brown", col="bisque", background="lightblue")
```



第 9 章

大気再解析

再解析データとは，比較的新しい予報モデルに対して過去の観測データを同化したものです。数値天気予報においては，予報モデルに対して観測を同化した解析値が作成され，初期値として用いられています。予報モデルは改良が重ねられていくため，解析値の品質はモデルの変遷と共に変化していきます。データ同化手法も大きく進歩し，数十年前とは全く異なる手法が用いられています。モデルやデータ同化手法の改良の影響を取り除くことにより，時間的により品質が安定したデータセットになります。再解析とは，現業予報で一度行った解析を再びやり直すということを意味しています。

9.1　ラスタデータ

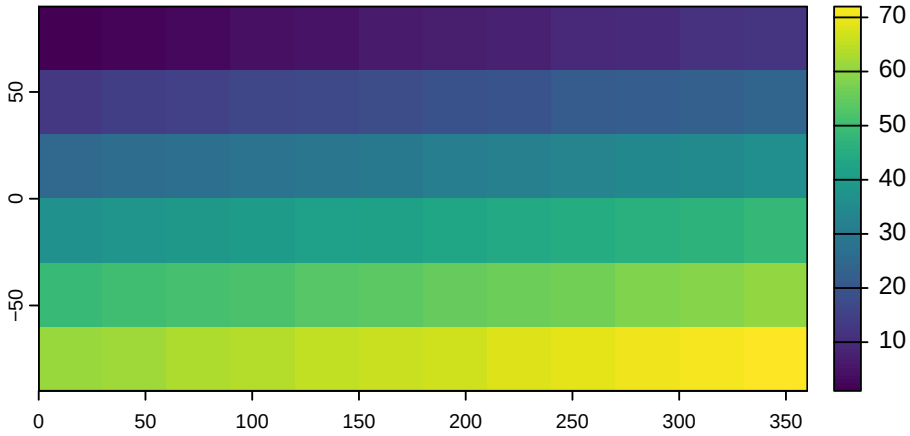
terra で格子点値は `SpatRaster` クラスで扱います。

SpatRaster にデータが格納される順序を確認します。

```
library(terra)
```

```
terra 1.8.5
```

```
r <- rast(ncol=12, nrow=6, xmin=0, xmax=360, ymin=-90, ymax=90)
values(r) <- 1:ncell(r)
plot(r)
```



列方向に連続する行優先で、列は西から東、行は北から南であることが分かります。セルに色が付いているので半格子ずれるが、ひとまず目をつぶることにします。

9.2　気候値

京都大学生存圏研究所生存圏データベースグローバル大気観測データでは、NCEP 再解析を提供しています。月平均/その他の統計処理済データ、surface とたどりましょう。

NCEP/NCAR 再解析データの海面気圧の月別気候値 slp.mon.ltm.nc を取得します。

NetCDF は、RNetCDF や ncdf4 で読むことができます。terra も `rast()` で NetCDF を読めることになっているが、上記ファイルは読めませんでした。

```
library(RNetCDF)

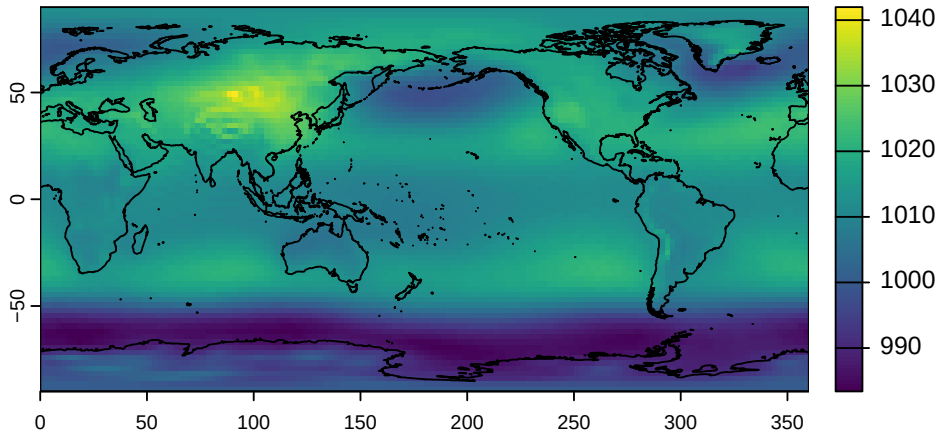
nc <- open.nc("slp.mon.ltm.nc")
slp <- var.get.nc(nc, "slp")
dim(slp)
```

```
[1] 144  73  12
```

変数は `var.get.nc()` で取得します。次元は経度、緯度の順で 144x73 です。R の配列は Fortran や MATLAB と同じ列優先（左の添え字が先に変わる）なので、`rast()` に渡すときは、転置 `t()` する必要があります。`SpatRaster` も北から南向きなので、南北は反転しません。データと経度を合わせて日付変更線を図の中心にするため、Chapter 8 で定義した `shift.lon()` を使っています。1 月の海面気圧を描きましょう。

```
#|echo: false
shift.lon <- function(v, dlon=0) {
  e <- crop(v, ext(dlon+0, 180, -90, 90))
  w <- crop(v, ext(dlon-180, dlon, -90, 90))
  w <- shift(w, 360)
  rbind(e, w)
}
```

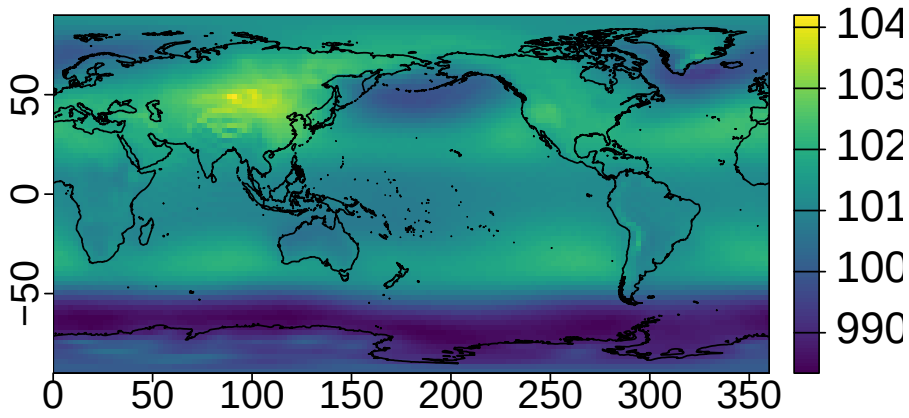
```
slp.ras <- rast(xmin=0, xmax=360, ymin=-90, ymax=90, ncols=144, nrows=73)
values(slp.ras) <- t(slp[,,1])
cshp <- "ne_50m_coastline/ne_50m_coastline.shp"
c50 <- vect(cshp)
c50 <- shift.lon(c50)
plot(slp.ras)
plot(c50, add=TRUE)
```



カラーパレット

軸のフォントサイズなどのパラメタは pax にリストとして指定します。同様にカラーバーのパラメタは plg に指定します。

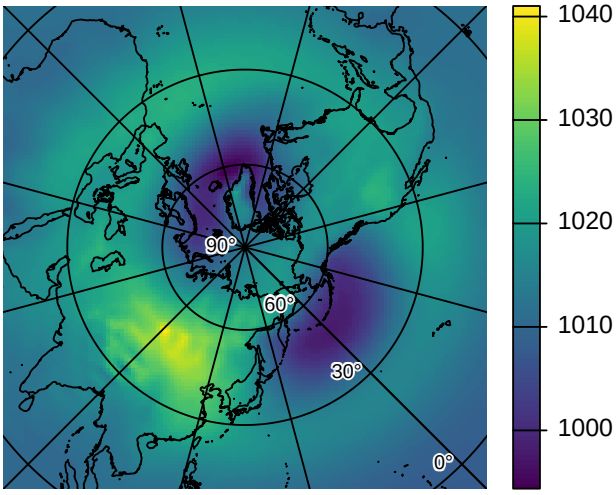
```
plot(slp.ras, pax=list(cex.axis=2), plg=list(cex=2))
plot(c50, add=TRUE)
```




```
slp.ras.c <- crop(slp.ras, ext(0, 360, -30, 90))

newcrs <- "+proj=stere +lon_0=135e +lat_0=90n"
c50p <- project(c50, newcrs)

slp.ras.p <- project(slp.ras.c, newcrs)
g <- graticule(30, 30, crs=newcrs)
plot(slp.ras.p, axes=FALSE, ext=ext(-1e+7, 1e7, -1e7, 1e7))
plot(c50p, add=TRUE)
plot(g, add=TRUE)
```



カラーパレット

色を変えるには、color.palette にカラーパレットを生成する函数を与えるか、col に明示的に色を与えます。col は color.palette に優先し、色の数はレベルより一つ少なくします。

- RColorBrewer など、様々なパレットがパッケージとして提供されています。
- NCL のカラーテーブルは rcolors に収められています。

9.3 月平均

NCEP 再解析の月平均/その他の統計処理済データ、surface から、今度は月平均データを取得します。ここでは層厚（1000 hPa と 500 hPa の厚さ）thickness.mon.mean.nc を選びました。

特定の年月に対応する番号は、1948 年 1 月が最初に入っているので、(y - 1948) * 12 + m で計算できます。

あるいは、日時に関する函数を使って次のように求めることができます。

```
library(RNetCDF)

nc <- open.nc("thickness.mon.mean.nc")
time <- var.get.nc(nc, "time")                                    ①
tunit <- att.get.nc(nc, "time", "units")                          ②
time.posixct <- utcal.nc(tunit, time, type="c")                   ③
ymd <- as.POSIXct(paste(c(1970, 1, 1), collapse="-"))             ④
t <- which.min(abs(time.posixct - ymd))                           ⑤
```

① 変数 time を取得します。
② 時刻の単位を取得します。
③ RNetCDF の utcal.nc() で POSIXct 型に変換します。
④ c() で作った年月日のベクトルを - で繋ぎ、POSIXct 型に変換します。
⑤ which.min() で最も小さい番号を求めます。

練習

同じ月について、最近の層厚と数十年前の層厚を比べてみよう。

第 10 章

前線形成

冷たい空気と暖かい空気とがぶつかると気温の勾配が大きな前線ができます。初期の気温の分布が次の式で与えられているとします。$\theta_0 = 290\mathrm{K}$ は参照温位、$\Delta\theta$ は温位の変動幅、d は傾圧帯の幅を表します。温位は定められた気圧にしたときの気温です。傾圧帯は気温の勾配がある帯状の領域です (Keyser et al. 1988)。

$$\theta_i = \theta_0 - \frac{\Delta\theta}{2} \tanh\left(\frac{n_i}{d}\right)$$

n_i は空気塊の前線に直交する座標で気温の勾配 $\nabla\theta$ の反対向きに取ります。添え字の i は初期時刻を表しています。

$$n_i = -x_i \sin\alpha_i + y_i \cos\alpha$$

空気がぶつかる風の場を表す流線函数は次のように表されます。

```
calc.pt <- function(x, y, alpha, theta0=290.0, dtheta=20.0, d=500.0) {
  n <- -x * sin(alpha) + y * cos(alpha)
  theta0 - 0.5 * dtheta * tanh(n/d)
}
```

$$\psi = -bxy$$

流線函数と風とは次のような関係があります。

$$u = -\frac{\partial\psi}{\partial y} = bx,\ v = \frac{\partial\psi}{\partial x} = -by \tag{10.1}$$

```
calc.psi <- function(x, y, b) {
  -b * x * y
}
```

風 Equation 10.1 を時間積分すると、空気塊の位置は時間とともに次の式で移動します。

$$x(t) = x_i e^{bt},\ y(t) = y_i e^{-bt}$$

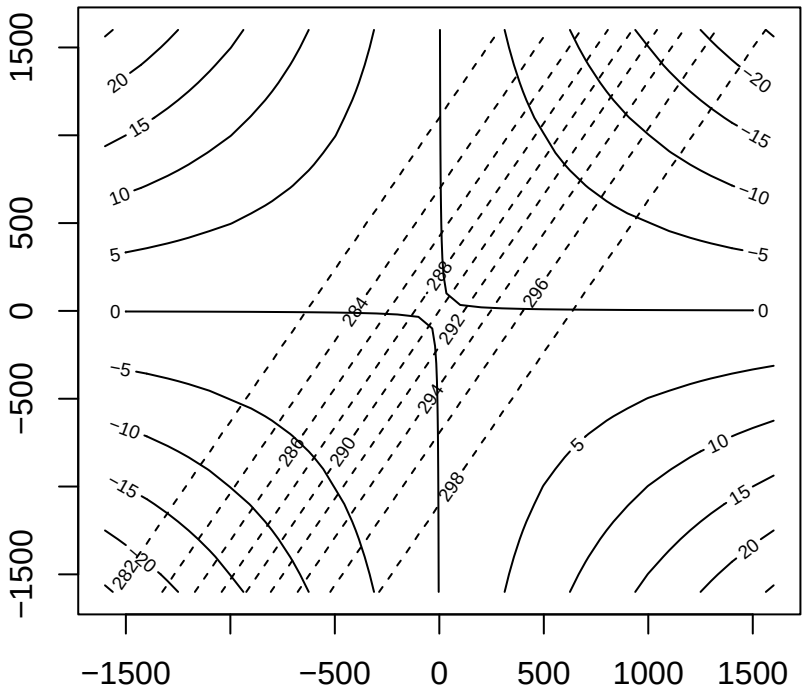
```
calc.xi <- function(x, b, t) {
  x * exp(-b * t)
}
calc.yi <- function(y, b, t) {
  y * exp(b * t)
}
```

南北に 3200 km の領域の設定します。

```
x <- seq(-1600, 1600, 100)
y <- seq(-1600, 1600, 100)
```

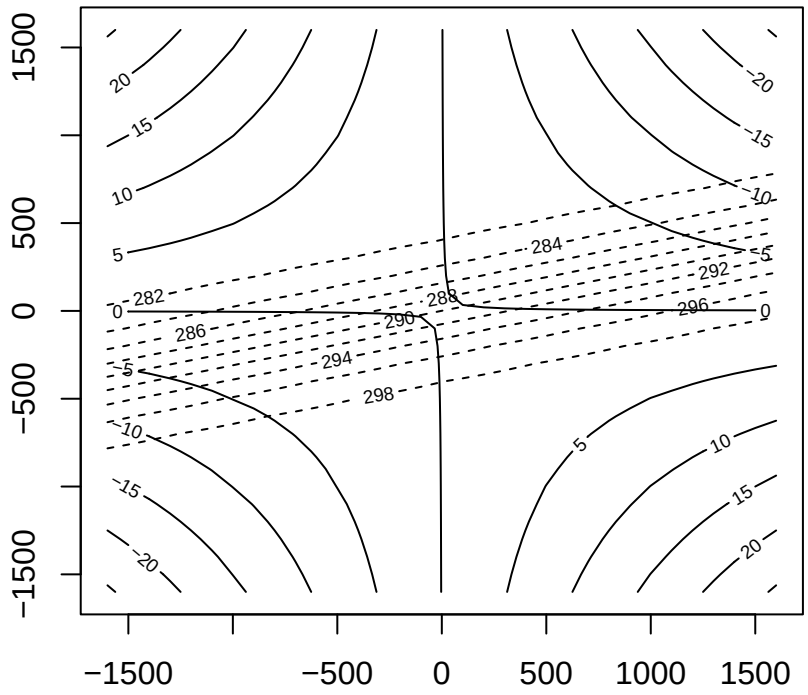
初期の流線函数と温位の分布は次のようになっています。

```
alpha <- pi / 3
b <- 1.0e-5
theta <- outer(x, y, calc.pt, alpha)
psi <- outer(x, y, calc.psi, b)
contour(x, y, psi)
contour(x, y, theta, lty=2, add=TRUE, asp=1)
```

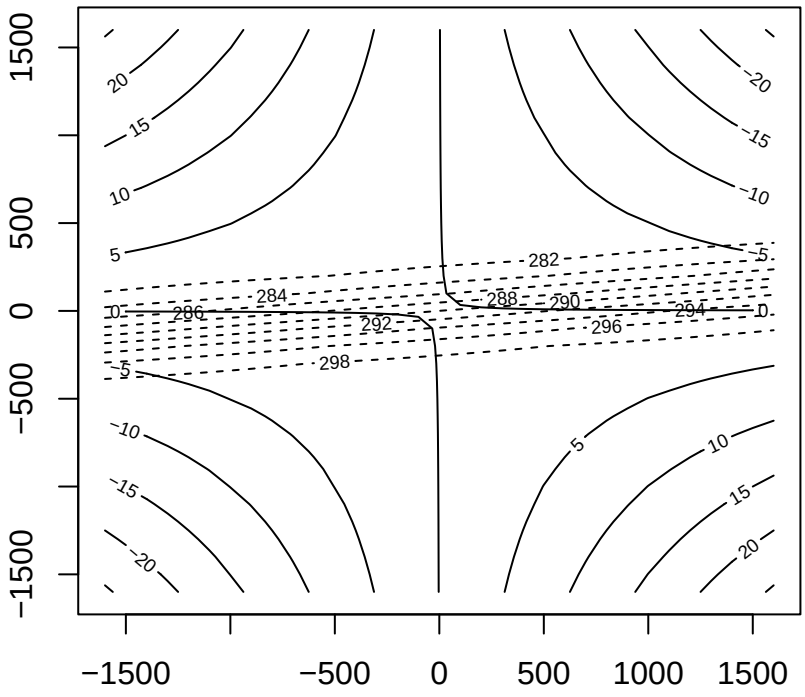


時刻 $t = b^{-1} = 27.8$ h、$t = 1.5b^{-1} = 41.7$ h と時間が進むにつれて、前線は時計回りに回転しながら勾配が強化されていきます。

```
alpha <- pi / 3
b <- 1.0e-5
t <- 1.0 / b
theta <- outer(calc.xi(x, b, t), calc.yi(y, b, t), calc.pt, alpha)
contour(x, y, psi)
contour(x, y, theta, lty=2, add=TRUE, asp=1)
```




```
alpha <- pi / 3
b <- 1.0e-5
t <- 1.5 / b
theta <- outer(calc.xi(x, b, t), calc.yi(y, b, t), calc.pt, alpha)
contour(x, y, psi)
contour(x, y, theta, lty=2, add=TRUE, asp=1)
```


1500
500
0
−500
−1500
−1500
−500
0
500
1000
1500
20
15
10
5
−20
−15
−10
−5
282
284
286
288
290
292
294
296
298

第 11 章

Lorenz-96 モデル

Lorenz and Emanuel (1998) は追加の観測をどこで行うと効果的か考察するために、次の簡単なモデルを用いました。

$$\frac{\mathrm{d}X_j}{\mathrm{d}t} = (X_{j+1} - X_{j-2})X_{j-1} - X_j + F$$

$j = 1, .., J$ は格子点の番号で、$J = 40$ がよく用いられます。格子点は緯度円上に等間隔に並んでいます。X は気温や渦度のようなスカラーの気象学的要素を表しています。右辺第 1 項は移流を表す非線型項で全エネルギー $(\sum_j X_j^2)/2$ は保存されます。右辺第 2 項は消散を表す線型項で全エネルギーを減少させます。係数が 1 になるようにスケールされていて、消散の時定数は 5 日です。強制項 F は、時空間平均 $\bar{X}$ の範囲 $[0, F]$、分散 σ の範囲 $[0, F/2]$ を決めます。$F > 8/9$ のとき波数 8 の波が成長します。このモデルの時間変化傾向は R で次のように書けます。

```
l96 <- function(x, F) {
  n <- length(x)
  (x[c(2:n, 1)] - x[c(n-1, n, 1:(n-2))]) * x[c(n, 1:(n-1))] - x + F
}
```

4 次の Runge-Kutta 法で時間積分をします。

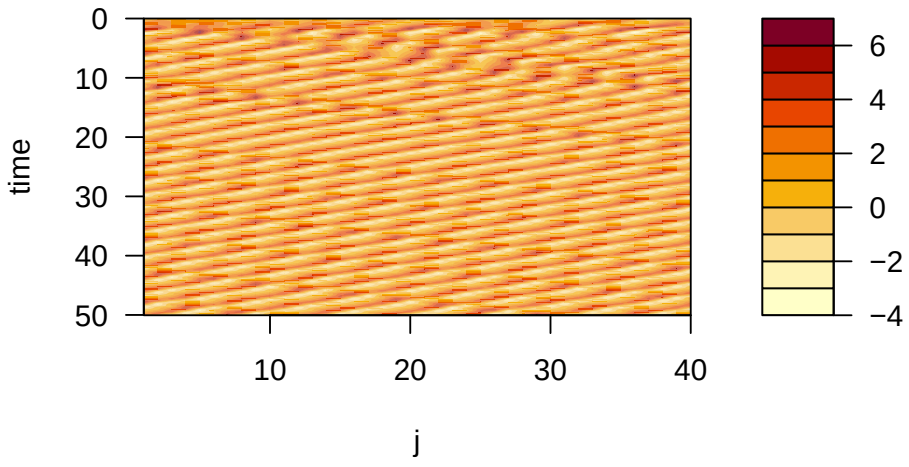
```
rk4 <- function(f, x, dt, opts) {
  k1 <- f(x, opts)
  k2 <- f(x + 0.5 * dt * k1, opts)
  k3 <- f(x + 0.5 * dt * k2, opts)
  k4 <- f(x + dt * k3, opts)
  x + (k1 + 2 * k2 + 2 * k3 + k4) * dt / 6
}
```

$F = 3.85$ で 1000 ステップ先まで時間積分します。時間刻み幅 0.05 は 6 時間に対応します。

```
nj <- 40
nstep <- 1001
x.hist <- matrix(0, nj, nstep)
x <- rnorm(nj)
F <- 3.85
dt <- 0.05
for (i in 1:nstep-1) {
  x <- rk4(l96, x, dt, F)
  x.hist[,i] <- x
}
```

等値線を描いてみます。

```
t <- seq(0, nstep*dt, length.out=nstep)
filled.contour(1:nj, t, x.hist, nlevel=11,
               ylim=rev(range(t)), xlab="j", ylab="time")
```



バイナリファイル

x をバイナリファイルに保存するには、writeBin() を使います。

```
fname <- file("l96.dat", "wb")
writeBin(x, fname)
```

計算とは別の描画スクリプトでバイナリデータを読むには readBin() を使います。

```
nx <- 40
nstep <- 100
x <- readBin("l96.dat", double(), n=nx*nstep)
```

縞々の数と向きから、波数は 8 で位相は時間とともに西に進んでいることが分かります。これに対し初期に振幅が大きな場所は時間とともに東に進んでいます。このようなふるまいは、定常解 $\bar{X} = F$ に対する摂動方程式の解で説明できます。

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = (x_{j+1} - x_{j-2})F - x_j$$

$X_j = \bar{X} + x_j$ とし摂動の 2 次の項は無視しました。波動解を仮定すると、$F > 8/9$ のときに波長 $L = 2\pi/k = 2\pi/\cos^{-1}(1/4) \approx= 4.767$ の波が成長することが示されます。波長はおよそ 5 なので、$J = 40$ に対しては波数は 8 に対応します。このとき不安定波の位相速度 $c = -(\sin k + 2\sin 2k)(F/k)$ はおよそ -1.09、群速度 $c_\mathrm{g} = -(\cos k + 2\cos 2k)F$ はおよそ $+1.17$ となります。このほかにも不安定モードが存在しており、$F > 2.0$ では波数 4 から 12 までの波が同時に不安定に、$F > 4.0$ では波の成長で強制項の効果を打ち消すことができなくなりカオスに遷移します。データ同化では $F = 8$ がよく用いられるようです。

練習

- F をいろいろと変えてみましょう。
- 平均や分散が理論どおりか確認してみましょう。
- 初期摂動の影響を調べてみましょう。試行のたびに結果はどのように変わるでしょうか。

第 12 章

機械学習

PyTorch は広く使われている機械学習フレームワークです。Torch for R は R から PyTorch のほとんどの機能が利用できます。

パッケージの名前は `torch` です。`install.packages("torch")` インストールしましょう。

12.1 数値最適化

Torch for R を解説した Keydana (2023) の Function minimization with L-BFGS に従って、数値最適化をしてみましょう。

最適化のベンチマーク函数 Rosenbrock 函数は

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

と書けます。x の 4 次函数となっており、歪んだ溝の中に最小値があります。そのため、最急降下法や共軛勾配法では、多くのステップ数を必要とします。ニュートン法やガウス・ニュートン法では、少ない回数で最小に到達することが知られています (Enomoto and Nakashita 2024)。

上記テキストでは、右辺第 2 項の係数が 5 になっていて、線型探索なしで 2 回、ありで 1 回で最適値に至るとしていますが、ここでは標準的な係数 100 を使います。

```
rosenbrock <- function(x, y, a = 1, b = 100) {
  (a - x)^2 + b * (y - x^2)^2
}
```

ここでは、torch for R を使って、Rosenbrock 函数を最適化します。最適化手法には、L-BFGS、線型探索には強ウルフ条件を使います。制御変数を torch のテンソルとして定義し、初期位置を与えるとともに、勾配を自動微分で求めるために `requries_grad=TRUE` を指定します。

```
library(torch)

x <- torch_tensor(c(-1, -1), requires_grad = TRUE)

optimizer <- optim_lbfgs(x, line_search_fn = "strong_wolfe")
```

損失を計算する函数を定義します。勾配を 0 に初期化して、損失とその勾配を計算します。最適化の進捗を確認するため、損失を表示しています。この函数は最適化手法の 1 ステップ optimizer$step() に渡します。

```
calc_loss <- function() {
  optimizer$zero_grad()

  value <- rosenbrock(x[1], x[2])
  cat("value is:", as.numeric(value), "\n")

  value$backward()
  value
}
```

3 ステップ数まで進み、損失を xhist に格納し、write() でテキストファイルに保存します。ファイルに格納せずに、直接描画しても構いません。

```
num_iterations <- 3
xhist <- as.numeric(x)
for (i in 1:num_iterations) {
  cat("\n", "iteration:", i, "\n")
  optimizer$step(calc_loss)
  cat("x=", as.numeric(x), "\n")
  xhist <- rbind(xhist, as.numeric(x))
}
```

```
 iteration: 1
value is: 404
value is: 62.32629
value is: 30.0694
value is: 2.630802
value is: 1.178554
value is: 1.15742
value is: 1.132393
value is: 1.00142
```

```
value is: 1.091282
value is: 0.6181912
value is: 0.8905501
value is: 0.6098283
value is: 0.5655912
value is: 0.3922533
value is: 0.2774411
value is: 1.417702
value is: 0.2190705
value is: 0.1747326
value is: 0.1380794
value is: 0.08087045
value is: 0.05257415
value is: 0.1490689
value is: 0.0400695
value is: 0.02954894
value is: 0.01238139
x= 0.9039377 0.8114879

 iteration: 2
value is: 0.01238139
value is: 0.006821597
value is: 0.002873288
value is: 0.001204705
value is: 0.0006028978
value is: 4.76946e-05
value is: 2.687239e-06
value is: 5.929914e-08
value is: 1.555293e-09
value is: 3.588241e-13
x= 0.9999999 0.9999998

 iteration: 3
value is: 3.588241e-13
x= 0.9999999 0.9999998
```

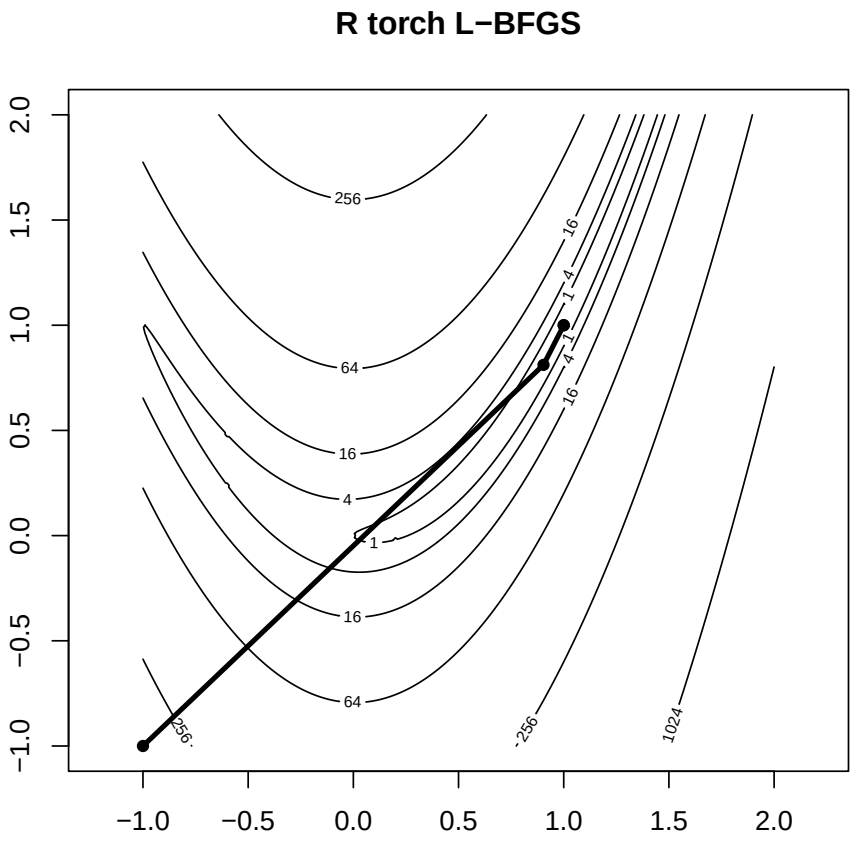
```
write(xhist, file = "hist.txt")
```

2 回で最小に至りました。テキストファイルに保存した最適化の履歴を読み、Rosenbrock の等値線に重ねて描画します。

```
xhist <- matrix(scan("hist.txt"), ncol = 2)

x.axis <- seq(-1, 2, 0.01)
y.axis <- seq(-1, 2, 0.01)
z <- outer(x.axis, y.axis, rosenbrock)

contour(x.axis, y.axis, z, levels=4^(0:10), main="R torch L-BFGS",
        asp=1, xlim=c(-1,2))
points(xhist[,1], xhist[,2], pch=16)
lines(xhist[,1], xhist[,2], lwd=3)
```



第 13 章

他の言語とのインターフェース

13.1　C と Fortran

SPHREPACK は球面調和函数ライブラリで、NCAR Classic Library for Geophysics から取得できます。

`gaqd.f` を使ってガウス緯度を求めてみましょう。C や Fortran を呼び出すには、まずソースから共有ライブラリを作ります。次のコマンドを実行すると、`gaqd.so` ができます。

```
% R CMD SHLIB gaqd.f
```

Base R には C や Fortran を呼び出すための FFI (Foreign Function Interfance) として、`.C()` と `.Fortran()` があります。これらは引数をあらかじめ用意する必要がある上、変数はコピーされるので効率が良くありません。ここでは、パッケージ dotCall64 を用います。dotCall64 は、64 ビットの整数を扱えるので、2GB 以上の配列を扱えます。

コンパイルされた函数を呼び出す `.C64()` は、C も Fortran も扱えます。引数の型を `double`, `integer`, `int64` から `SIGNATURE` に指定します。`INTENT` には、read（読み込み）、read and write（読み書き）、write（書き込み）を意味する `r`、`rw`、`w` が指定できます。- `r` は変数を用意する必要がありますが、コンパイルされた函数にポインタ（メモリの場所）が渡されるだけで、コピーはされません。- `rw` は R オブジェクトのコピーが生成され、コンパイルされた函数はポインタを受け取ります。- `w` に対応する変数は、`vector_dc()`、numeric_dc()、integer_dc()‘で割り付けることができます。これらの仕組みにより、R オブジェクトの生成を制御できます。

```
library(dotCall64)

dyn.load("gaqd.so")

gaqd <- function(nlat) {
  w <- 0
  lwork <- 0L
```

```
  gaus <- .C64("gaqd",                                                    ①
       c("integer", "double", "double", "double", "integer", "integer"), ②
       nlat, theta = numeric_dc(nlat), wts = numeric_dc(nlat),           ③
       w, lwork, ierror= integer_dc(1),
       INTENT=c("rw", "w", "w", "r", "r", "w"))                          ④
  gaus[c(1, 2, 3, 6)]
}
```

① サブルーチンの名前を渡します。
② SIGNATURE として引数の型を指定します。
③ ガウス余緯度 theta と重み wts は numeric_dc() で長さ nlat の倍精度浮動小数点数ベクトルを作っています。エラーコード ierror は長さ 1 の整数ベクトルを作っています。
④ 返り値に nlat を含めるため、nlat は rw で指定しています。theta と wts、w と lwork は、ダミー変数なので r とします。

13.2 Rcpp

Rcpp を使うと、R から C/C++ のコードを使えます。RcppParallel: は並列プログラミング、RcppEigen や RcppArmadillo は線型代数演算を扱う C++ ライブラリ Eigen と Armadillo の機能を提供します。

13.3 Python

reticulate を使うと、Python を R から呼び出すことができます。

例えば、Numpy の npy ファイル x.npy を読むには次のようにします。

```
library(reticulate)

np <- import("numpy")
x <- np$load("x.npy")
```

第 14 章

プレゼンテーション

Quarto を使うと R のコードや実行結果を埋め込んだレポートやプレゼンテーション、ウェブページを作成することができます。ここでは、ウェブブラウザで表示する Revealjs プレゼンテーションを作ってみよう。

タイトルや著者、作成するドキュメントの種類はヘッダと呼ばれる部分に書きます。文書の中身は Markdown という形式で書きます。

- スライドの見出しの前には ##、箇条書きは - で並べます。
- 文中の数式は $ で別行立ての数式は $$ で囲み、LaTeX の記法で書きます。
- R のコードと結果を入れることもできます。

```
---
title: タイトル
author: 名前
format: revealjs
---

## スライドの見出し

- 箇条書き
- 次の項目

## 数式

$$
g(z) = \frac{GM}{(a+z)^2}
$$

## R のコード

```{r}
```
```

```
df <- read.csv("co2_annual_20221026.csv")
plot(df$year, df$co2.global.mean.ppm., main="Global Mean CO2 concentration", xlab
```
```
```

References

Anscombe, F. J., 1973: Graphs in Statistical Analysis. *The American Statistician*, **27**, 17–21, https://doi.org/10.2307/2682899.

Bolton, D., 1980: The computation of equivalent potential temperature. *Mon. Wea. Rev.*, **108**, 1046–1053, https://doi.org/10.1175/1520-0493(1980)108%3C1046:TCOEPT%3E2.0.CO;2.

Enomoto, T., and S. Nakashita, 2024: Application of exact Newton optimisation to the maximum likelihood ensemble filter. *Tellus A*, **76**, 42–56, https://doi.org/10.16993/tellusa.3255.

Evenden, G. I., 1991: *Cartographic projection procedures for the UNIX environment—a user's manual*. United States Department of the Interior Gelological Survey, https://download.osgeo.org/proj/OF90-284.pdf.

Keydana, S., 2023: *Deep learning and scientific computing with r torch*. Chapman & Hall/CRC Press,.

Keyser, D., M. J. Reeder, and R. J. Reed, 1988: A Generalization of Petterssen's Frontogenesis Function and Its Relation to the Forcing of Vertical Motion. *Monthly Weather Review*, **116**, 762–781, https://doi.org/10.1175/1520-0493(1988)116%3C0762:AGOPFF%3E2.0.CO;2.

Lorenz, E. N., and K. A. Emanuel, 1998: Optimal sites for supplementary weather observations: Simulation with a small model. *J. Atmos. Sci.*, **55**, 399–414, https://doi.org/10.1175/1520-0469(1998)055%3C0399:OSFSWO%3E2.0.CO;2.

赤穂昭太郎, 2008: **カーネル多変量解析—非線形データ解析の新しい展開**. 岩波書店,.