

文系のための R セミナー

第 1 回 R をはじめてみよう：ベクトル操作まで(6/23(火), 26(金))

R 概要 5 分

- R のメリット

環境構築：Posit Cloud の登録 10 分

簡単な計算 5 分

- 画面構成の解説
- スクリプトファイルの作成
- 簡単な計算

変数とオブジェクト 10 分

- オブジェクト作成

ベクトル作成と簡単な操作 10 分

- ベクトルとは
- ベクトルの計算
- ベクトルの簡単な操作

スクリプトファイルの保存 5 分

演習問題(時間があれば)

第 2 回 データの可視化をしてみよう(7/3(金), 7(火))

環境構築：Posit Cloud の登録

クラスと関数

データフレームの操作

データのビジュアライゼーション

散布図とスクリプトファイルの保存

演習問題

R 概要

Rとは、統計解析やデータ可視化に特化したプログラミング言語で、無償で利用できます。現在人気のあるPythonなどの汎用プログラミング言語と比べると、アプリ開発などには向いていないものの、回帰分析や時系列分析、仮説検定など統計の基礎から高度な手法まで幅広く対応しています。エクセルでも基本的な回帰分析はできますが、Rを使うことで、テキストデータや空間データの分析も扱うことができます。またエクセルは表計算ソフトですので、直感的に使うことができますが、自分が行った作業の過程が少し分かりにくいところがあります。他方、Rはコードを書くという行為が挟まる分、手間はかかりますが、分析の過程が記録され作業の工程をあとから確認することができます。さらに同じ作業を別のデータ分析に再利用することができます。

	R	Excel	Python
使用タイミング	統計分析、学術研究	幅広い業務、教育統計	データ分析、Web・AI開発
費用	無料	有料	無料
特徴	・統計分析に特化 ・グラフや図の可視化が簡単 ・文章で残しやすい	・直感的な操作 ・マクロが使える	・ライブラリが豊富 ・機械学習やAI分野で主流
習得難易度	慣れるのに少し時間がかかる	簡単	直感的
汎用性	○ 統計分析のみ	△ 統計や機械学習は苦手	◎ 統計・機械学習・Web開発など幅広く対応
グラフ作成	◎ 簡単・キレイに作成可能	△ 簡単に作図可能。ただし微調整が難しい	○ Rに比べるとやや難しい

なお、回帰分析とは計量分析におけるもっとも基本的な手法です。ある変数 Y と、ある変数 X の関係を確かめるために、二つの変数の散布図を作成し、その上に最も適した直線を引くことで、2つの変数の関係を一次関数的に近似する手法です。

環境構築：Posit Cloud の登録

Rはインストールすれば単体でもすぐに分析することができます。しかし、コードが書きにくかったり、統計分析のためのデータの読み込みが大変だったりします。なので、一般的には**RStudio**と呼ばれるソフトウェアをダウンロードして利用します。このようなソフトを**IDE(Integrated Development Environment、統合開発環境)**と呼びます。RStudioはRをより便利に使うためのソフトで、これから行う分析をより簡単に行うことができます。

プログラミング言語と聞くと、親しみにくいイメージがある方も多いと思います。しかし、文法や用語は完璧に覚える必要はなく、適宜教科書等を参照することで慣れていくことが大切です。IDE は言語の学習と実践的な利用をサポートしてくれる環境であり、これを活用すればより効率的に R でデータ整理・分析を行うことができます。RStudio を用意するなど、R を始めとしたプログラミング言語を自分のパソコンで利用できるようにすることを環境構築といいますが、使用しているパソコンやネット環境などによっては、とても時間がかかる場合があります。よって、今回は **Posit Cloud** (<https://posit.cloud>) というサービスを利用します。

Posit Cloud とは、RStudio をブラウザ上で利用することができるサービスで、RStudio を管理している団体が運営しています。サービスに多少の制限はありますが、無料で利用できるのも、今回はこちらを利用します。まず、Posit Cloud のサイトにアクセスし、画面右上の「Log in」を選択します。

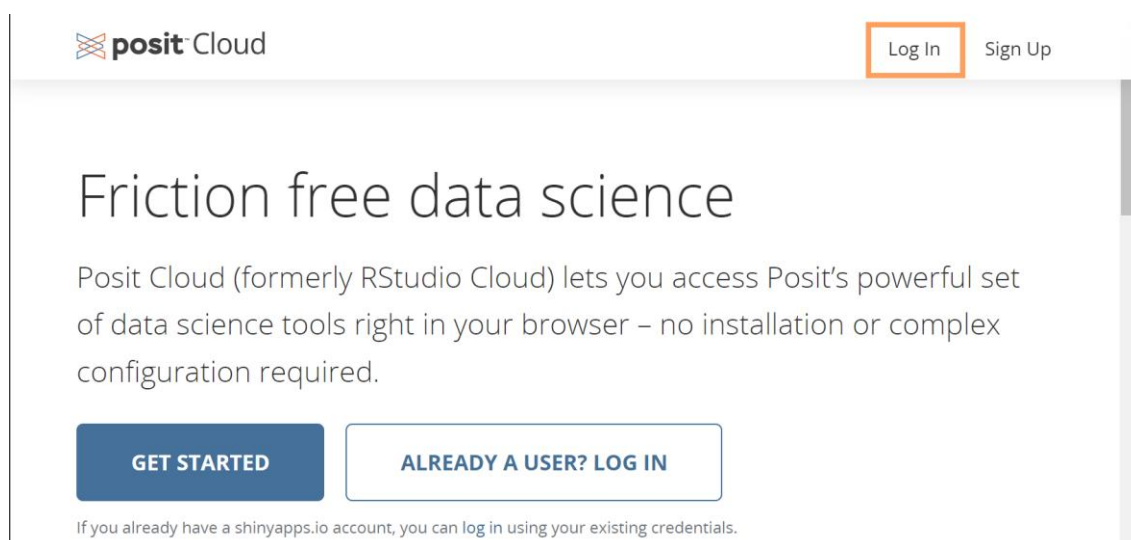


図 1. Posit Cloud のトップページ

次に「Sign Up」を選択し、画面上部に必要情報を入力する、もしくは、Google アカウントを持っている方は下の「Sign Up with Google」から手続きを進めてください。Google アカウント以外で登録(Sign up)を行った場合は、差出人” noreply@posit.cloud” から件名” Please verify your email address” というメールが届きます。メール本文中の” Verify your email” と書かれたリンクをクリックすると、承認完了です。

図 2. 利用登録(サインアップ)画面

登録が完了し、ログインができれば、図 3 のような画面が表示されるはずです。

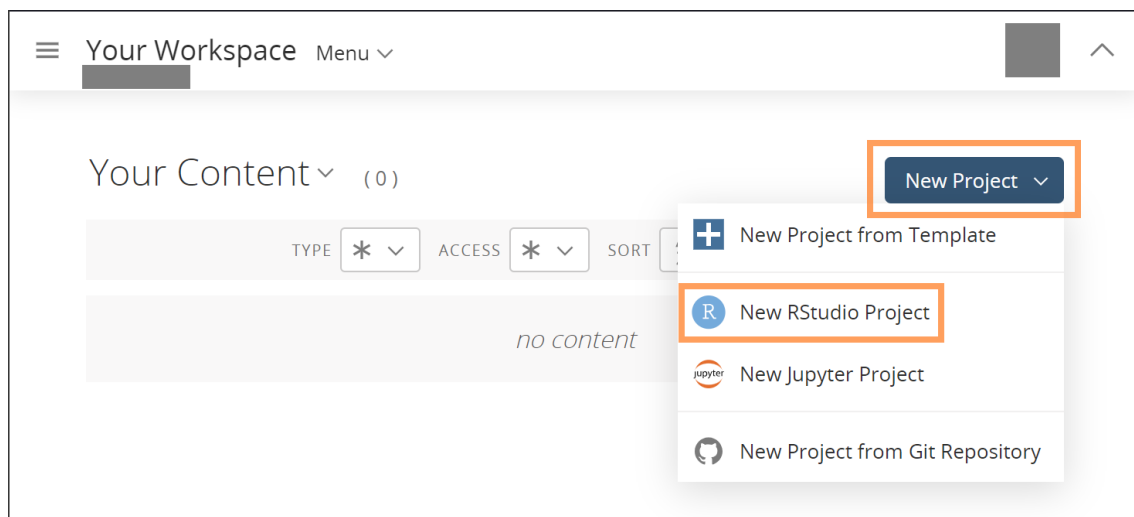


図 3. プロジェクト選択

画面右上の「New Project」を選択し、表示されたリストから「New RStudio Project」を選択してください。(次の画面の立ち上げまで多少時間がかかります)
その後、次のような画面が表示されるはずです。

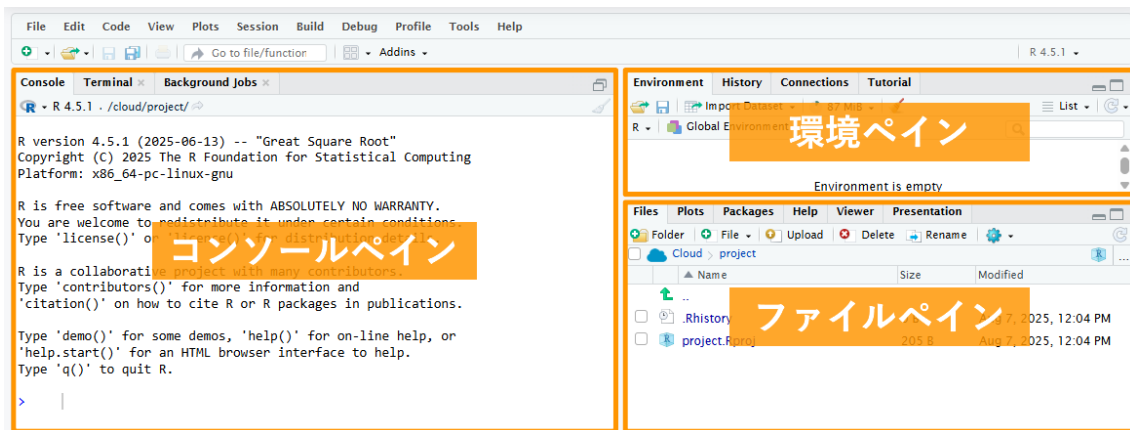


図 4. RStudio の基本画面

RStudio のウィンドウは、**ペイン (pane)** と呼ばれるいくつかの領域から成り立っています。それぞれの役割については都度、解説していきます。

これで今回の環境構築は完了です！続いて簡単な操作を試してみましょう。

簡単な計算

ここでは簡単なコードを実行して、その手順を確認してみましょう。まず、コードを入力するための場所である、**スクリプト (Script)** を作成します。画面左上の緑のプラスマークをクリックし、「R Script」を選択して下さい。すると、左上のエリアに新しいペインが表示されたはずです。これを**ソース (Source)** ペインといいます。ここにコードを入力して作業していきます。

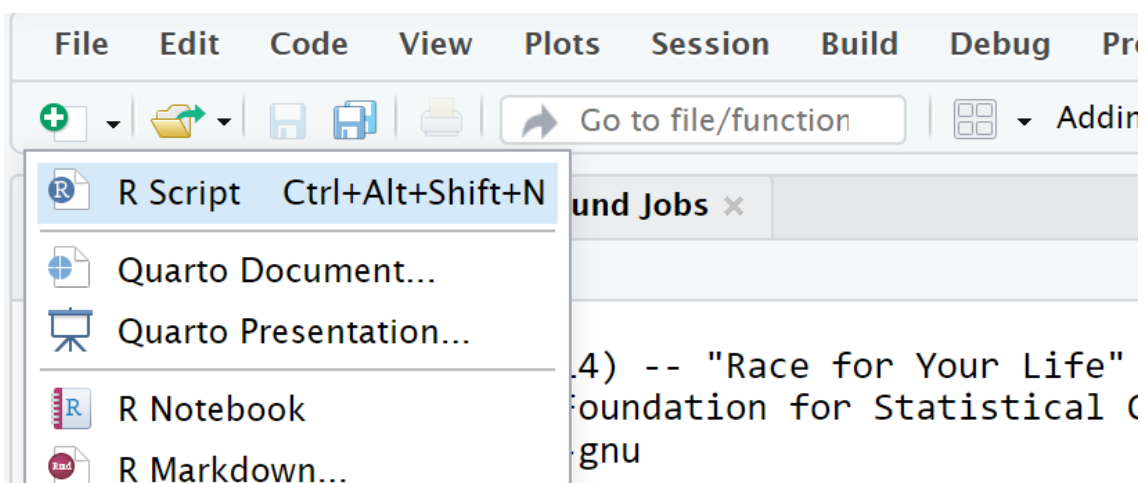


図 5. R スクリプトの新規作成

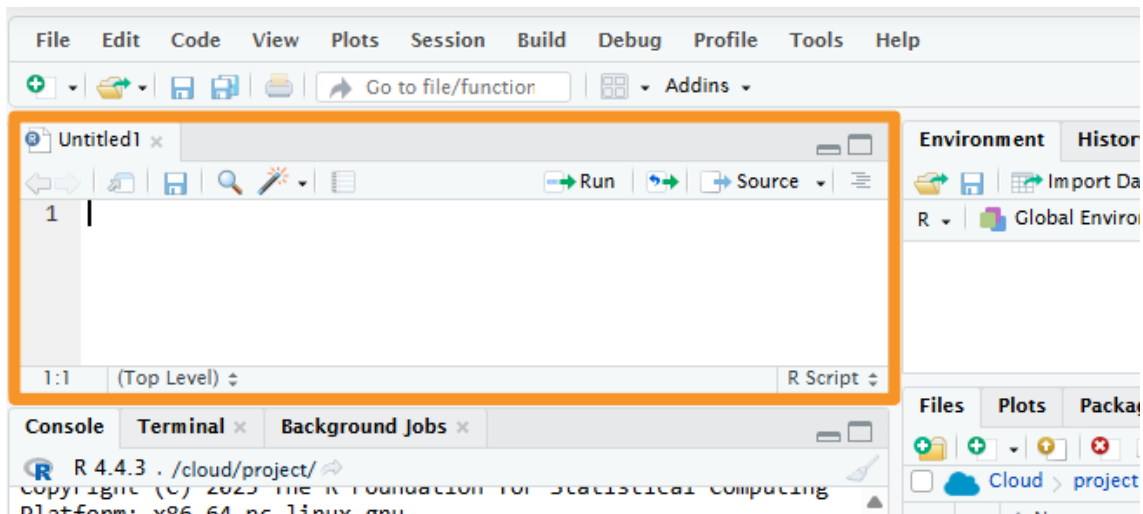


図 6. ソースペイン(R Script の入力画面)

R を実行するための準備が整ったので、コードを入力してみましょう。

まず、キーボードの入力モードが半角になっているか確認してください。次に、下のコードをスクリプトに入力し、コードを書いた行にカーソルを合わせて、**Ctrl + Enter**(Macの場合は **Command + Return**)を押下して実行してみましょう。なお、下記の#(ハッシュ記号)の右側部分はコメントです。コードの実行に直接関係ありませんが、自分の作成したコードを人と共有したり、みずから後で見返したりするときに、何を表しているコードかを一目で確認できるように記述されます。これ以降、補足としてコメントが挿入されていますが、皆さんがコードを実行する際はこの部分は入力する必要はありません。

4 + 7 # 以降、この部分にコメントを随時入れています。

[1] 11

上記の「##[1]」の右側がコードの実行結果です。Posit Cloud 上の表示形式と異なるかもしれませんが、結果が一致していれば問題ありません。Posit Cloud 上で実行結果は、コンソール(Console)ペイン(画面左下)に表示されます。

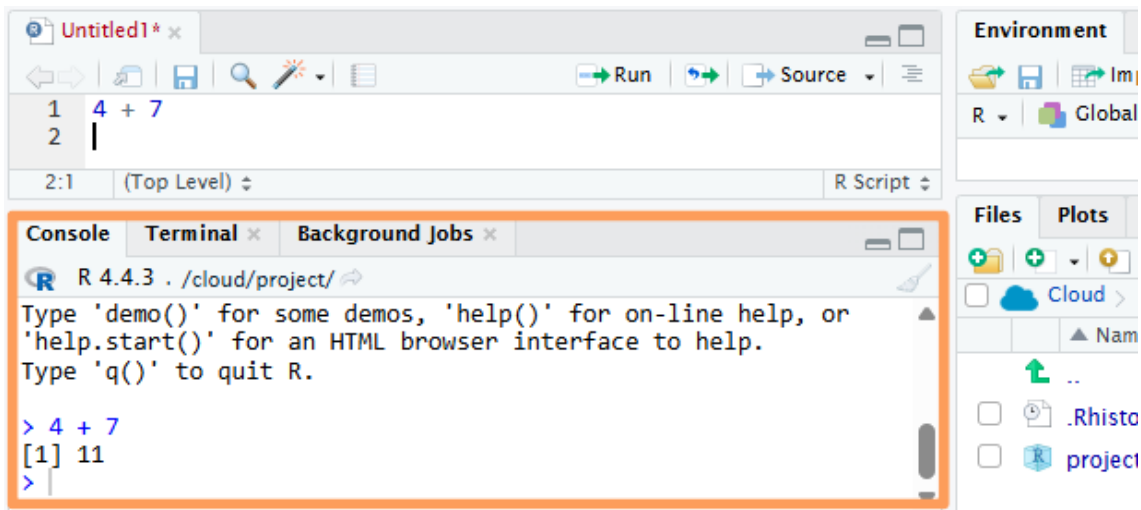


図 7. コンソールペイン

コードをよく見ると、加算演算子(+)の前後は半角スペースを空けていますが、半角スペースを空けなくてもコードは実行可能です。短いコードだと、あまり気にならないかもしれませんが、長いコードになると見間違いが生じやすくなるため、基本的には+や-などの演算子(operator)の前後は半角スペースを空けましょう。

```
3+4
```

```
## [1] 7
```

```
7 - 3
```

```
## [1] 4
```

```
8 / 3
```

```
## [1] 2.666667
```

```
6 ^ 3 # ベキ乗。日本語キーボードであれば、右上あたりの「^」の位置です。
```

```
## [1] 216
```

```
5 * (10 - 3) # 括弧の前後にスペースは要りません。
```

```
## [1] 35
```

変数とオブジェクト

もう少しコードらしいものを実行してみましょう。Rで扱うデータには、10や3.14のような数値だけでなく、大阪大学や大学生のような文字列もあります。このような具体的なデータそのものを **値(value)** といいます。Rでは、数値、文字列、ベクトル、データフレーム、関数など、Rの中で扱えるものは広い意味で**オブジェクト(object)** と呼ばれま

す。たとえば、10 という数値も、“大阪大学” という文字列も、c(1, 2, 3)というベクトルも、R ではオブジェクトとして扱うことができます。計算のたびに同じ値やデータを毎回入力するのは大変です。そこでR では、オブジェクトに名前を付けて保存することができます。この名前のことを**変数(variable)**と呼びます。たとえば、x という名前に 10 という値を保存しておけば、その後は x と入力するだけで 10 を使うことができます。変数に値を保存することを代入または割り当てといいます。R では、代入には主に割り当て演算子(<-)を使います。

「<-」の左側に変数名。右に代入したいもの

```
result <- 5 + 3 # 変数は基本的には半角英数字で作成
```

```
result # 変数を実行することで変数に入れた値を表示する
```

```
## [1] 8
```

作成した変数は画面右上の**環境(Environment)**ペインでいつでも確認することができます。

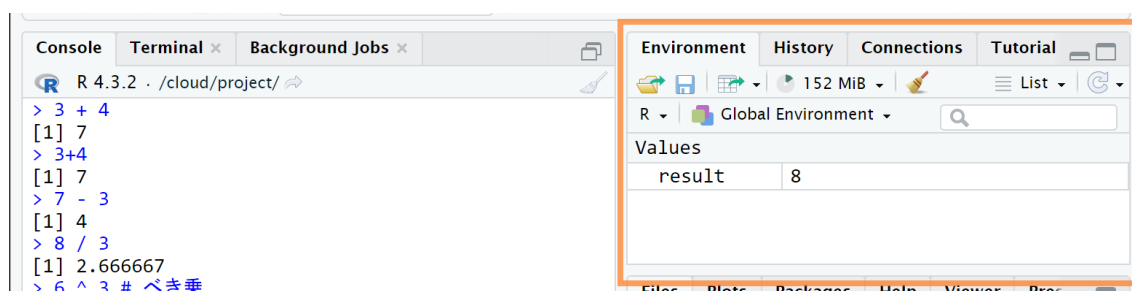


図 8. 環境ペイン

数値だけでなく、文字列も変数に入れることができます。その場合「“(二重引用符)”」か「'」(単一引用符)」で入れたい文字列を囲みます。引用符はどちらを使ってもかまいませんが、前後同じもので囲んでください。また、値である文字列と変数は混同しやすいですが、引用符で囲まれたものが値としての文字列になるのに対して、引用符で囲んでないものは変数としての文字列です。

```
daigaku <- "The University of Osaka" # 「"」はキーボードの「2」の位置にあります。
```

```
daigaku
```

```
## [1] " The University of Osaka"
```

数値を引用符で囲めば、文字列として扱うこともできます。ただし、その場合、先ほど行ったような演算はできなくなります。

```
Result <- "5"
```

```
Result
```

```
## [1] "5"
```

```
Result / 3
```

```
## Error in Result/3: non-numeric argument to binary operator
```

ベクトル作成と簡単な操作

前節で学んだ概念に加えて R の特徴の一つであるベクトルという概念について学びます。R では、「24」や「The University of Osaka」のような一個だけのデータだけではなく、複数のデータを 1 つにまとめたベクトルという形で変数に入力することもできます。例えば、1 つのベクトルに「1, 2, 3」や「apple», “banana”, “cherry」のようなデータを格納できます。

R におけるベクトルとは、同じ種類のデータ(文字列や数値など)で構成された、一次元の配列のことをいいます。なので、先ほどまでに作成したオブジェクトはすべて、長さ 1 のベクトルと言うこともできます。早速、ベクトルを作成する基本的な方法のひとつである、`c()`関数で作ってみましょう(c は連結の意味(concatenate)の頭文字)。

```
vec1 <- c(1, 2, 3) # コンマの後ろは半角スペース
```

```
vec1
```

```
## [1] 1 2 3
```

ベクトルとしてまとめることで、まとめて計算することが可能になります。

```
vec1 * 3
```

```
## [1] 3 6 9
```

```
vec1 + 10
```

```
## [1] 11 12 13
```

```
vec2 <- c(4, 5, 6)
```

```
vec1 + vec2
```

```
## [1] 5 7 9
```

変数名や関数名など何度も入力すると間違ってしまうことも増えてきます。そのようなミスを減らすために、RStudio では予測変換(補完機能)が使えます。下図のように、途中で入力すると、候補が出てくるので、複数ある場合は矢印キーで目的のものを選び、Tab キーや Enter キーで選択してください。

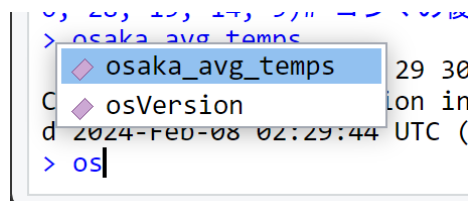


図 9. 補完機能

ベクトルに含まれる個々の値を**成分**と言います。成分のひとつひとつには**インデックス**という番号が振られており、これを使って特定の値を抜き出したり、一部の値だけを変更したりすることができます。先ほどの `vec1 + vec2` のような計算は同じ位置の成分同士で演算されています。

```
vec1[1] # ベクトル名[インデックス]で指定した値を取り出すことができます。
## [1] 1
var1 <- vec1[2] # 取り出した値を別の変数に入れることもできます。
var1
## [1] 2
vec1[2] #ベクトルを直接変更したわけではないので、元のベクトルの値はそのままです。
## [1] 2
```

また、ベクトルは一定範囲の値を抜き出すこともできます。

```
vec3 <- c(2, 4, 6, 8, 10, 12)
vec3[2:4]
## [1] 4 6 8
```

そして、ベクトル同士を連結することもできます。

```
vec4 <- c(vec1, vec2)
vec4
## [1] 1 2 3 4 5 6
```

スクリプトファイルの保存

最後にスクリプトファイルの保存をしましょう。最初に述べたように、R のメリットの 1 つは分析した手順をスクリプトファイルとして保存し、それを再利用できることです。のちに見返したときにわかるようにコメントをつけて残しておきましょう。ファイルの先頭に次の内容を書き込んでおくと、見返したときに便利です。

- ファイル名
- ファイルを作った目的、分析内容
- 作成者
- 作成日
- 最終更新日(または更新日のリスト)と更新者の名前



図 10. コメント例

入力出来たら Ctrl(Command) + S で保存し、実際のファイル名とコメントのファイル名を合わせておきましょう。今回、Posit Cloud で保存したファイルはクラウド上にあるので、今回作ったファイルが欲しい場合は、ファイルペインでファイルにチェックを入れ、「(歯車)More」→「Export…」でダウンロードしてください。

演習問題(時間があれば)

問 1.

`vec <- c(10, 20, 30, 40, 50)` というベクトルがあるとき、「20」と「40」を取り出して、新しいベクトル `vec_sub` を作ってみましょう。

問 2.

以下のような文字列ベクトル `fruits` から “banana” のみを取り出すコードを書いてください。

```
fruits <- c("apple", "banana", "cherry")
```

問 3.

`c()`関数を使わずに `1:10` のような方法で、連続する数値ベクトルを作ってみましょう。

問 4.

以下のように長さが違うベクトル同士を足し算した場合、どのような結果になるか確認してください。

```
a <- c(1, 2, 3)
b <- c(10, 20, 30, 40)
a + b
```

問 5.

次のような数値と文字が混ざったベクトルを足し算した場合、どのような結果になるか確認してください。

```
v1 <- c(1, 2, "osaka")
```

```
v2 <- c(3:5)
```

```
v1 + v2
```

```
## Error in v1 + v2: non-numeric argument to binary operator
```