synQSL seminar

10. Aug. 2024

Hands-on Programming Workshop

CONTENTS：

作成者：

増谷　佳孝（yoshitaka.masutani.a8@tohoku.ac.jp）
東北大学・医学系研究科・保健学専攻・画像情報学分野
JSMRM スタディグループ「生成型学習による定量的 MR イメージング」代表

STEP.0　synQSL-kit-2408 に含まれる各種ツール・データ群の説明

（1）データ合成ソフトウェア（C/C++）

場所：synQSL-kit-2408/dMRIsynthLite

・C/C++のソースコードおよび makefile プロジェクト：　dMRIsynthLite

　入力：MPG ファイルおよびコマンドラインからのオプション

　出力：標準出力（csv ファイル）

（2）回帰器（MLP）用のツール群（python）

場所：synQSL-kit-2408/python_codes

・MLP の訓練用コード：　MLP_train_csv_args_2407.py

　入力：　csv 形式の合成データ（正解を含む）およびコマンドラインからのオプション

　出力：　keras 形式の MLP データ

・MLP のテスト（検証）用コード：　MLP_test_csv_args.2407.py

　入力：　keras 形式の MLP データおよび csv 形式のテスト用合成データ（正解を含む）

　出力：　推定結果およびテスト結果（誤差などのサマリ）の csv データ

・MLP による予測用コード：　MLP_pred_csv_args.2407.py

　入力：　keras 形式の MLP データおよび csv 形式の推定用データ

　出力：　推定結果の csv データ

（3）その他のツール群（python）

場所：synQSL-kit-2408/python_codes

・NIfTI 形式(.nii)を csv 形式に変換する python コード：　nii2csv_args.py

　入力：　NIfTI 形式のボリュームデータ

　出力：　csv 形式の原信号値データ

・csv 形式の原信号を対数信号減衰比に変換する python コード：　makeLogDecay_args.csv

　入力：　csv 形式の信号値データ（原信号）

　出力：　csv 形式の信号値データ（対数信号減衰比）

・csv 形式を RAW 形式に変換する python コード：　csv2raw_args.py

　入力：　csv 形式の数値データ

　出力：　RAW 形式のボリュームデータ

・RAW 形式のマスク処理を行う python コード：　maskRaw_args.py

　入力：　RAW 形式のボリュームデータ

　出力：　RAW 形式のボリュームデータ

（4）シェルスクリプト

場所：synQSL-kit-2408/shell_scripts

・DKI モデルの作成済み合成データを使った一括学習処理： train_all_DKI1D_regressors

・DTI モデルのデータ合成～サンプル画像での推定処理： do_all_DTI3D

（5）データ

場所：synQSL-kit-2408/data/dwidata/

・NIfTI 形式の DWI データ（DKI 用, DTI 用）： sampleDWI.DKI1D.nii, sampleDWI.DTI3D.nii

・マトリクス：64x64, スライス数：64(DKI), 39(DTI)

・RAW 形式のマスク（同上）：sampleDWI.DKI1D.nonair_mask.raw, sampleDWI.DTI3D.brain_mask.raw

場所：synQSL-kit-2408/data/mpg

・dTV-II 形式の MPG データ（DKI 用, DTI 用）： sampleMPG.DKI1D.txt, sampleMPG.DTI3D.txt

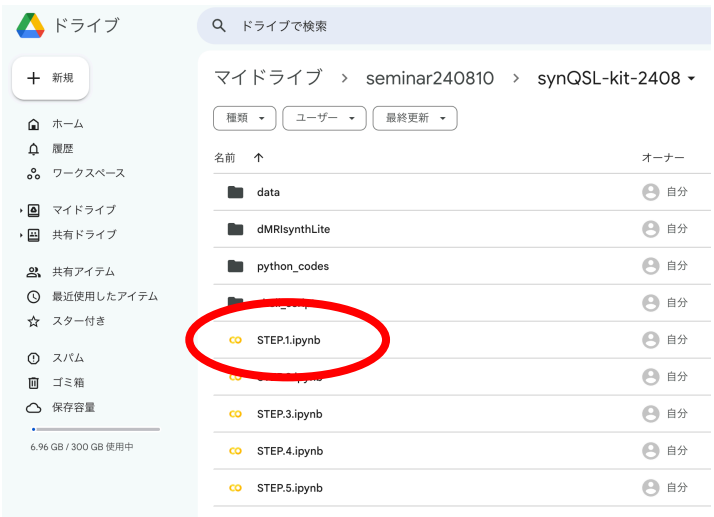
※その他に空のフォルダが synQSL-kit-2408/data フォルダ内にありますが消去しないでください

（6）Google ドライブおよび Google Colaboratory（Google Colab あるいは Colab と略）

・Google アカウントでログイン後、Google ドライブ画面にて下記の手順を行う

1. Google ドライブ画面にて、マイドライブの直下に解凍した seminar240810 フォルダをコピー

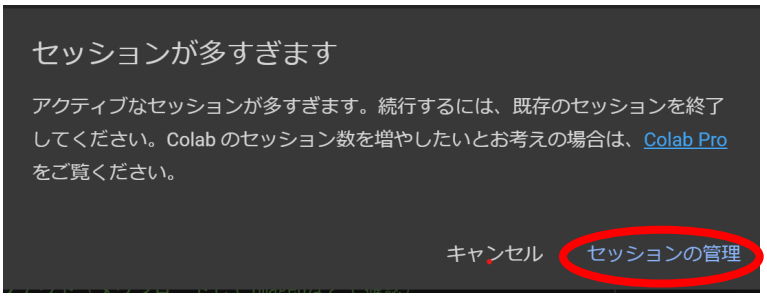
2. seminar240810/synQSL-kit-2408 フォルダ内のノートブックをダブルクリックして開くと、Colab 画面でノートブックが表示される



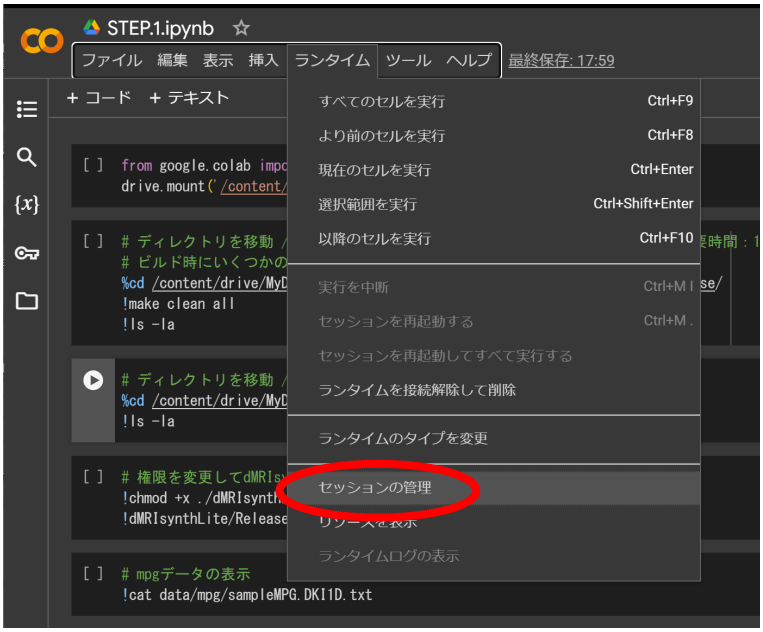
※ ノートブックを開いた直後の作業について

・どのノートブックも最初はマイドライブをマウントするスクリプトのセルがあるので、実行ボタン「▶」（下図◯）を押す。その後、アカウントの選択などの確認を経てマウントされる。

・作業終了したノートブックは閉じてもランタイムのセッションが残っている場合があり、GPU 使用量の制限から下記のようなメッセージが出る場合がある

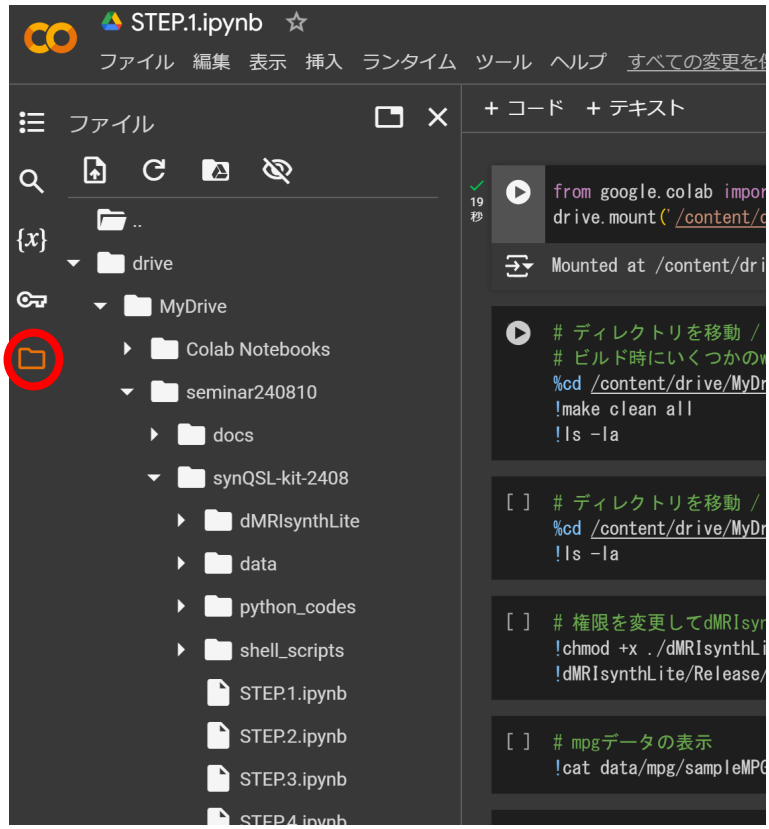


・上記の場合、「セッションの管理ボタン」を押すか、あるいはノートを閉じるときにあらかじめ Colab の「ランタイム＞セッションの管理」メニュー（下図）から以下の管理ツールを呼び出して、使用するノートブック以外のセッションを終了する（ノートブックを閉じるときにこまめにセッションを終了することを推奨）



・セッション管理ウィンドウの各セッションの右端の Trash ボタン（下図◯）で終了させる。

・その他、画面右にファイルブラウザの on/off ボタン（下図◯）があり、一部のファイル（テキストなど）はダブルクリックで開いて内容を確認できる



STEP.1 データ合成

内容：dMRIsynthLite によるデータ合成

・Colab のノートブック（STEP.1.ipynb）を使用

・Google Colab の shell でビルドと実行を行う（ソースコードの説明および改造は STEP6 で）

・各実行オプションの設定はコマンドラインから（順を追って説明）

ポイント

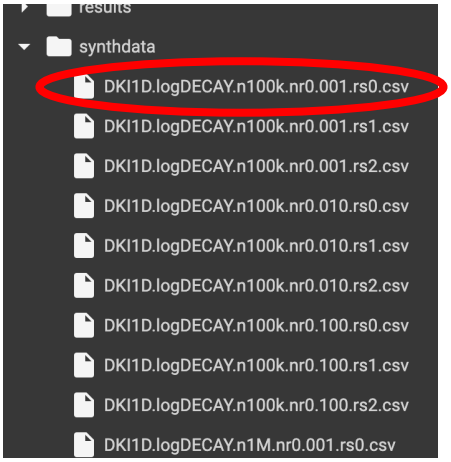
・UNIX の基本コマンドに関する知識が必要（付録１）

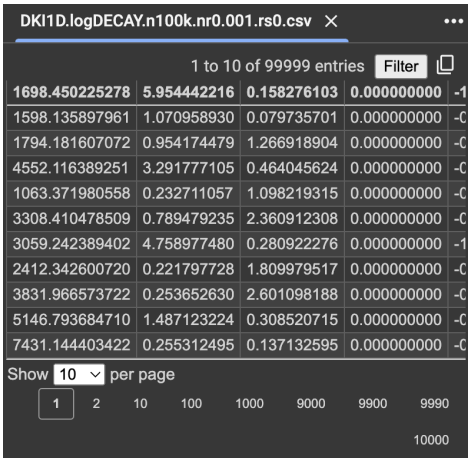
・対数信号減衰比の入力信号を3種のノイズ比、3種のシードで作成する

・実行ログを残したい場合は、「編集」メニューから「ノートブックの設定」で、「このノートブックを保存する際にコードセルの出力を除外する」のチェックを外す

・作成したデータの確認

・ファイルブラウザから csv ファイルをダブルクリックして確認可能





STEP.2 ノートブックによる回帰器の学習とテスト

内容：回帰器の学習とテスト

・Colab のノートブック（STEP.2.ipynb）を使用

・ノートブックで python コードを逐次実行

・モジュールの import

・各種設定

・学習データの読み込み

・回帰器の定義

・回帰器の構築と学習

・回帰器のサマリ表示

・学習のヒストリ表示（学習曲線）

・学習データによる RMS 誤差評価

・回帰器モデルの保存

・回帰器モデルを再読み込みして検証・テスト（RMS 誤差評価）

ポイント

・GPU を使用するために、「ランタイム」メニューの「ランタイムのタイプを変更」において、
「T4GPU」を選択してから実行（GPU の使用には時間制限があるので注意→付録2参照）

・学習処理中に表示されるのは MSE loss なので RMSE の2乗の値

・回帰器モデルの学習では、繰り返して性能の出るものを選ぶ必要がある

・何度か繰り返して学習の初期化により性能が変わることを確認（最適化の初期値問題）

・loss や RMSE でだいたい同じような値 → 何度か繰り返すと悪い／良い結果が出る

STEP.3　コマンドラインからの python コードファイルの実行

内容：コマンドラインからの python コードファイルの実行による回帰器の学習とテスト

・Colab のノートブック（STEP.3.ipynb）を使用

・コマンドラインで”!python（コード名）（オプション）”と入力して学習とテストを実行

ポイント

・ノイズ比 0.010 での学習データでの RMSE の目安

・拡散係数 Dx1k：　0.05～0.06

・拡散尖度 K:　 0.06～0.07

STEP.4　シェルスクリプトの利用

内容：シェルスクリプト実行による回帰器の学習とテスト

・Colab のノートブック（STEP.4.ipynb）を使用

・STEP.3 でのコマンドライン入力を複数記述したシェルスクリプトを実行

ポイント

・最下部のコマンドライン入力（x2）は、下記の目安に達しない場合の人力チューニング用

・ノイズ比 0.001 での学習データでの RMSE の目安

・拡散係数 Dx1k：　0.03 程度

・拡散尖度 K:　 0.025 程度

・ノイズ比 0.010 での学習データでの RMSE の目安

・拡散係数 Dx1k：　0.06 程度

・拡散尖度 K:　 0.1 程度

・ノイズ比 0.100 での学習データでの RMSE の目安

・拡散係数 Dx1k：　0.37 程度

・拡散尖度 K:　 0.31 程度

STEP.5　画像データにおけるパラメタ推定

内容：実画像でのパラメタマップ推定

・ImageJ で NIfTI 形式の DWI データ（nii ファイル）を確認

・Colab のノートブック（STEP.5.ipynb）を使用

・DKI のデジタルファントムデータの処理を下記のステップごとに実行

・NIfTI 形式の DWI データを csv に変換

・原信号から対数信号減衰比に変換

・コマンドラインから推定（予測）処理

・推定結果の csv を RAW 形式に変換

・RAW 形式で空気領域の除外を行うマスク処理

・DTI の実データ処理

・学習データ合成、モデル構築（パラメタ x2，ノイズ比 x3で計6種）、データ変換、推定処理、RAW 形式のパラメタマップ生成、マスク処理を一括して行うシェルスクリプトを実行（5～6分）

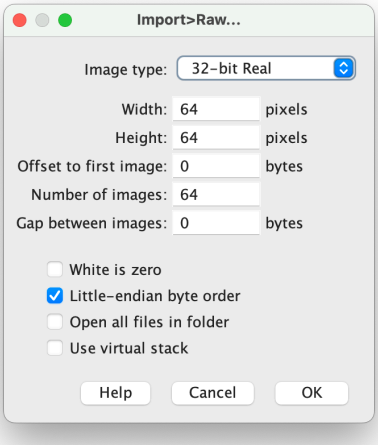
ポイント

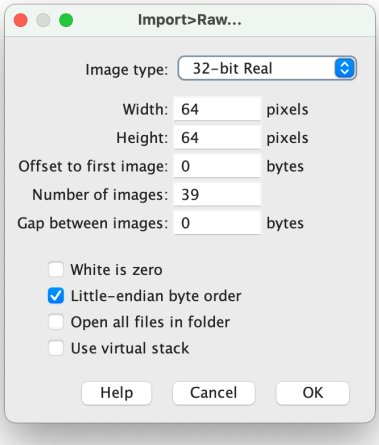
・結果は元データ（nii ファイル）と同じ場所に保存される

・結果確認は RAW 形式のデータをダウンロードして ImageJ で読み込んで行う

・Google ドライブから dwidata のフォルダごとダウンロードが簡便（圧縮もしてくれる）

・ImageJ のツールバーにドラッグ&ドロップして下記の設定を入力（左：DKI 用、右：DTI 用）





STEP.6　データ合成における新規モデルの追加（FWI モデル）

内容：dMRIsynthLite の概要を理解し、他の信号値モデル（FWI）を導入する

・プロジェクトに含まれるファイル・フォルダ構造の概要と内容（付録3参照）

・Colab のノートブック（STEP.6.ipynb）を使用

・FWI3D モデルを追加して再ビルド

ポイント：

・新しい信号値モデルに対応する3つのファイルを作る

XXX.params.h：　XXX.params.cpp のみで使用するパラメタのレンジの定義

XXX.h：　信号生成関数の定義（dMRIsynthLite_main.cpp にインクルードする）

XXX.params.cpp：　信号生成関数の実際の記述

・main 関数を含む dMRIsynthLite_main.cpp を編集する

・XXX.h をインクルードする

・新しい信号値モデルが指定された時の記述を追加する

・3つのファイルをプロジェクトに追加してビルドする

実際の手順（FWI3D モデルの例）：

1. SignalModels フォルダ内に3つのファイルが作成済み

・FWImodel3D.h, FWImodel3D.params.h, FWImodel3D.cpp

2. source/dMRIsynthLite_main.cpp を編集（バックアップをとっておくこと）

・FWImodel3D.h をインクルード

```
else
if (!strcmp(argv[1], "FWI3D"))
{
    data_generation_by_FWImodel3D( num_total_channels, num_So_channels, b_values, g_vectors,
                                   num_samples,
                                   noise_ratio, signal_type, with_noiseless, show_progress);
}
```

・show_usage_and_exit 関数の model_type 表示に追加した新規モデル名「FWI3D」を追加

・"model_type: DKI1D, or DTI3D"→"model_type: DKI1D, DTI3D, or FWI3D"

3. ビルド

・visual studio などの統合環境でビルドおよびデータ作成を行う場合

作成した3つのソースコードをプロジェクトに追加してビルドを行う

・データ合成を Colab 上で実行する場合（STEP.1 と同様）

下記の場所および名称の make ファイルを編集（バックアップをとっておくこと）してビルドする

（STEP.6.ipynb でビルドして実行・確認）

dMRIsynthLite/Release/source/SignalModels/subdir.mk

既存のモデルの記述に倣って計4箇所に FWI の記述追加（下図の赤字部分）

```
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
CPP_SRCS += \
../source/SignalModels/DKImodel1D.cpp \
../source/SignalModels/DTImodel3D.cpp \
../source/SignalModels/FWImodel3D.cpp

CPP_DEPS += \
./source/SignalModels/DKImodel1D.d \
./source/SignalModels/DTImodel3D.d \
./source/SignalModels/FWImodel3D.d

OBJS += \
./source/SignalModels/DKImodel1D.o \
./source/SignalModels/DTImodel3D.o \
./source/SignalModels/FWImodel3D.o


# Each subdirectory must supply rules for building sources it contributes
source/SignalModels/%.o: ../source/SignalModels/%.cpp source/SignalModels/subdir.mk
	@echo 'Building file: $<'
	@echo 'Invoking: GCC C++ Compiler'
	g++ -O3 -Wall -c -fmessage-length=0 -MMD -MP -MF"$(@:%.o=%.d)" -MT"$@" -o "$@" "$<"
	@echo 'Finished building: $<'
	@echo ' '


clean: clean-source-2f-SignalModels

clean-source-2f-SignalModels:
	-$(RM) ./source/SignalModels/DKImodel1D.d ./source/SignalModels/DKImodel1D.o \
		./source/SignalModels/DTImodel3D.d ./source/SignalModels/DTImodel3D.o \
		./source/SignalModels/FWImodel3D.d ./source/SignalModels/FWImodel3D.o

.PHONY: clean-source-2f-SignalModels
```

※ makefile もフォルダ構造に従って複数のファイルで階層構造を持つことに注意（特に subdir.mk はフォルダごとに複数あり、内容が異なるので注意）

※ eclipse により生成された makefile プロジェクトであるため「Do not edit」との記載があるが編集して OK、ただし eclipse で読み込む際は注意

まとめ・全体のポイント

・実画像での推定では学習とテストのデータでのノイズ量の一致が重要

実際には実画像でのノイズレベル推定は容易ではないので、異なるレベルの複数モデルを学習しておき、視覚的あるいは定量的に評価してどのレベルが良いかを判定する

・性能を少しでも向上させるには

(1) 同じハイパーパラメタでも繰り返し(初期値が異なる)によるトライがある程度必要
⇒1 回だけだと、たまたま性能の悪いモデルを拾うことがある

(2) 実際にはハイパーパラメタのチューニングを含め、シェルスクリプトで全自動化することが望ましい(次回)

(3) データサンプルの量も重要(今回は 10^5⇔10^6 が標準)
⇒大量のデータ生成に時間がかかるので並列化も重要(次回)

・Google Colab 以外での実行

(1) 臨床画像を使用する場合、独自環境で実行する必要がある
⇒csv ファイル化すると簡単にはわからないが、ハッキングは容易

(2) 独自環境の場合、tensorflow などの各種モジュールのバージョンが異なると正常動作しないことがあるため、なるべく現状の Colab と近い環境の整備を推奨するが、Colab もアップデートがあり対象が必要になることも

・Google Colab の時間制限については付録2(制限緩和には有償版)

・mpg ファイルおよび撮影データの制限

・dMRIsynthLite では対数信号減衰比の使用を推奨しており、その場合は元データおよび mpg ファイルにおいて b=0 の画像が最初に1つだけあることを仮定している(下図)

```
dMRIsynthLite(.exe) model_type mpg_filename [options]
        model_type: DKI1D, or DTI3D
        mpg_file: dTV-II format
        output: stdout is default. Add > filename at the end to save as a CSV file.
        options:
                -signal_type label: output signal type (RAW, DECAY, logDECAY: default is RAW)
                -with_noiseless label: output noiseless signal (true or false: default is false)
                -noise_ratio value: ratio to mean So signal level (default is 0.0)
                -num_samples value: number of samples (default is 1)
                -random_seed value: seed for random number generator (default is 0)
                -show_progress label: show percentage of progress (true or false: default is false)
        note:
                * random number generation is by mt19937-64 (see mt19937-64 folder)
                * currently single b=0 MPG file is supported
                * Do NOT use old powershell in Windows, which does NOT output in UTF-8
        command example:
                > dMRIsynthLite(.exe) DKI1D myMPG.txt -num_samples 100 -noise_ratio 0.1 > DKI1D.n100.nr0.1.csv
```

・bvec/bval ファイルを dTV-II 形式に変換する場合には上記の制限に注意

・HCP データなど、b=0 の画像が複数ある場合は平均 b=0 画像の作成を推奨

（付録1）LINUX の主要コマンドと Google Colab での実行について

・主要コマンド

ls -la	現在のディレクトリ（フォルダ）内のあるファイルの名前を表示する。-la オプションはサイズ、作成日時などの詳細も表示する。
cd ディレクトリ名	相対指定では現在のディレクトリ内のディレクトリを指定、または一つ上の階層（“cd ../”）を指定する。ちなみに現在のディレクトリは”./”。 絶対的な指定は、最上層＝ルートディレクトリ（/から始まる）から指定
pwd	現在のディレクトリを表示
rm ファイル名	ファイルの消去
chmod	権限の変更を行う。読み書きや実行の許可を行う ・実行形式の権限付与の例： chmod +x ファイル名（シェルスクリプトやプログラム）
mkdir 新規ディレクトリ名	ディレクトリの作成
man コマンド名	コマンドのヘルプ
プログラム（シェルスクリプト）名	プログラムなどを実行（ただし実行権限が必要）

・Google Colab での実行

セルの中に”!”、または”%”を入力してから、LINUX のコマンドを入力（複数可能）してスクリプト実行ボタンを押す。両者の違いは以下の通り。

! コマンド： コマンドでのディレクトリ移動などはセル内にのみ反映される

% コマンド： コマンドでのディレクトリ移動などはセル外にも反映される

ノートブックでディレクトリを移動しながら作業を進める場合、”%cd ディレクトリ名”を使用する

（付録2）Google Colab の時間制限について

Colaboratory

よくある質問（https://research.google.com/colaboratory/faq.html?hl=ja より）

Colab ではノートブックはどのくらいの時間動作しますか？

Colab では、インタラクティブ コンピューティングが優先されます。アイドル状態の場合、ランタイムはタイムアウトします。

料金がかからないバージョンの Colab の場合、ノートブックは可用性と使用パターンに応じて最長で 12 時間実行できます。Colab Pro、Pro+、従量課金制では、コンピューティング ユニットの残量に応じてコンピューティングの可用性が高くなります。

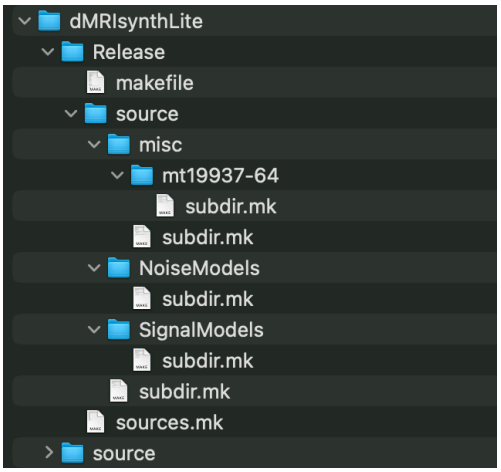
ノートブックは通常、可用性と使用パターンに応じて最長で 12 時間実行できます。Pro、Pro+、従量課金制プランでは、利用可能なコンピューティング ユニットを使い切るとバックエンドが終了する可能性があります。

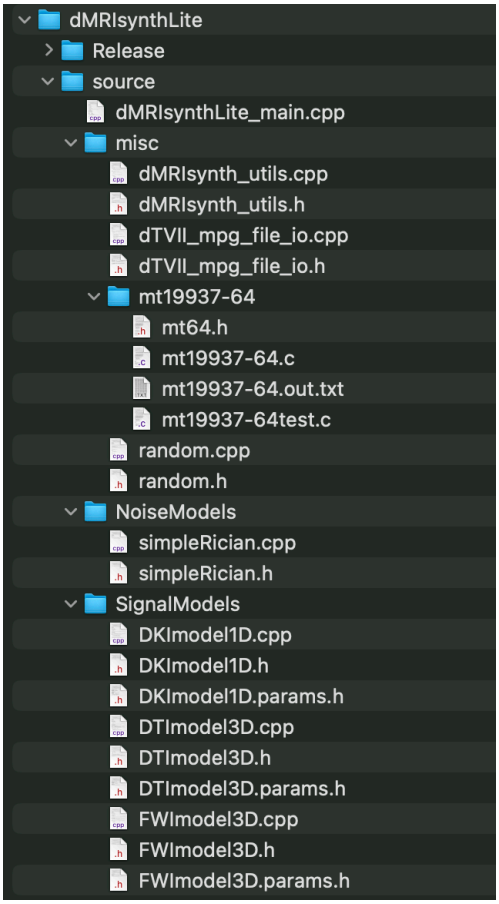
Colab Pro+ では、十分なコンピューティング ユニットがあれば、最長 24 時間連続でコードを実行できます。アイドル タイムアウトは、コードの実行が終了した場合にのみ適用されます。

GCP Marketplace で専用の VM を購入すると、ランタイムの制限とアイドル タイムアウトを完全に緩和できます。

（付録3）dMRIsynthLite の makefile プロジェクトのファイル構造

・ビルド前のフォルダ・ファイル構造（下図左：Release フォルダ内／下図右：source フォルダ内）





・source/misc/mt19937-64 フォルダおよび内容物は、メルセンヌツイスターによる乱数発生関連のコードであり、下記の URL より入手した

http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/mt64.html

・ビルド前の Release フォルダは、ビルド時に使用する階層化された makefile 群（makefile および各サブディレクトリ内の subdir.mk）を含むが、ビルドに従って object ファイル（*.o） などが生成され、実行可能な dMRIsynthLite 本体も Release 直下に生成される