This folder contains the following: a README file, folders, and Python scripts

[output](https://drive.google.com/open?id=1bTjADM8_P5v6nuHTARnUzzS0c83ZMS7P&usp=drive_copy)

[Testing](https://drive.google.com/open?id=1f_N7DFfmS-KvZWCHB7erge1-WHsdyHFu&usp=drive_copy)

[Testing\_cod\_for\_the\_chemical\_fraction](https://drive.google.com/open?id=16Cz7plPCj742-vHY6I2PZbU-_OOlkV_p&usp=drive_copy)

[Data\_Ana\_2.py](https://drive.google.com/open?id=1T7XOzMq7C8FlI4VffPiD4-uOxE5bRY7a&usp=drive_copy)

[Data\_Analysis\_1.py](https://drive.google.com/open?id=16Qk64IHK3Nr-cBctbnByVTAhr_BxQyYd&usp=drive_copy)

[just\_in\_case.py](https://drive.google.com/open?id=1AyZtm4W--MzTIDoDDZrvr74GbOhVQDDd&usp=drive_copy)

[Koyukuk\_river.py](https://drive.google.com/open?id=1kiIgVnEwwU1ghGpLFlsyO49MK_rAhkaY&usp=drive_copy)

[node\_headings.py](https://drive.google.com/open?id=1Rs6O2FszWCmdcvQIrEASTCQguFPeDIFN&usp=drive_copy)

[Pelly\_river.py](https://drive.google.com/open?id=15HsP-hc6XRqMiBCfpwQqmJgDOE-_451n&usp=drive_copy)

[Porcupine\_river.py](https://drive.google.com/open?id=1-OSNoNLG12lRJo62Pv9lKhNCmYFPomb7&usp=drive_copy)

[README](https://drive.google.com/open?id=1HrXuqYjeXIJt52-Di4Ttbp4nmFBbsYFGKtmuQ_WNgjk&usp=drive_copy)

[river\_initial\_monac\_code.py](https://drive.google.com/open?id=1KeLZASRwez9wyzOW_u6jOodW7U4TRuD7&usp=drive_copy)

[Stewart\_river.py](https://drive.google.com/open?id=1uN2ttk1iKRAaSLF5C2dT8wOVqtnKKH7x&usp=drive_copy)

[Tanana\_river.py](https://drive.google.com/open?id=1sq-OmMf7DRo1E9aJxVdEVOBvQPluRjJH&usp=drive_copy)

[Teslin\_river copy.py](https://drive.google.com/open?id=1CpVc2hTtx33MS7ZOllsMNlp_GBg7gGbn&usp=drive_copy)

[Teslin\_river\_plots.py](https://drive.google.com/open?id=1V9XKN_NNd5U63SRhUtVxWWUy8cw8IiRY&usp=drive_copy)

[Teslin\_river.py](https://drive.google.com/open?id=1yZtmOod1TNqkFkuEH37wTpxt1AmiQutp&usp=drive_copy)

[Test\_0.py](https://drive.google.com/open?id=1Ww10YG9C3dNUEq3kz7k3hWkYEuPpjwZp&usp=drive_copy)

[White\_Donjec\_river.py](https://drive.google.com/open?id=12XpmUGZWS8ZsQ-gNQe0Ys9qh-XgmXYfP&usp=drive_copy)

[Yukon\_river\_plot.py](https://drive.google.com/open?id=1KpQeUl_bpUXS2IKa5E2-9J2hjWxGJbHI&usp=drive_copy)

[Yukon\_river.py](https://drive.google.com/open?id=1gwh1bZR3DvVEJkI_GKR9tk5js1Ggqi_F&usp=drive_copy)

\*\*\*\*\*\*\*\*

Readme file

[README](https://drive.google.com/open?id=1HrXuqYjeXIJt52-Di4Ttbp4nmFBbsYFGKtmuQ_WNgjk&usp=drive_copy)

Folders

[Output](https://drive.google.com/open?id=1bTjADM8_P5v6nuHTARnUzzS0c83ZMS7P&usp=drive_copy)

The output folder contains an analysis folder and separate folders for the Yukon River tributaries and the main Yukon River stem.

[Testing](https://drive.google.com/open?id=1f_N7DFfmS-KvZWCHB7erge1-WHsdyHFu&usp=drive_copy)

Testing code on how random values are generated for the Monte Carlo analysis.

[Testing\_cod\_for\_the\_chemical\_fraction](https://drive.google.com/open?id=16Cz7plPCj742-vHY6I2PZbU-_OOlkV_p&usp=drive_copy)

Testing code on how random values are generated for the Monte Carlo analysis for the chemical fraction. This fraction code doesn’t obey the conservation law, which tells that the sum of the fractions should equal one.

Python Codes

[River\_initial\_monac\_code.py](https://drive.google.com/open?id=1KeLZASRwez9wyzOW_u6jOodW7U4TRuD7&usp=drive_copy)

This code allows the user to manually adjust the average values and boundaries (minimum and maximum) of the input parameters and also model parameters. It generates random values for each parameter within the specified boundaries. The number of simulations can be controlled by changing the num\_samples parameter. Also, it contains codes that create distribution plots for each input parameter that show their distribution types, max and min boundary values that are distributed around their mean values. For some distribution functions, code has been written to truncate negative input parameter values.

The outputs of this script are,

[CDOM\_component\_fractions.csv](https://drive.google.com/open?id=1e-25TrWO72cwrdX1J8fnobwZYm--_sUf&usp=drive_copy)

[Chemical\_fraction.csv](https://drive.google.com/open?id=1QlpA8AgiG62V_nM_A1RRJSf8VGUFHhda&usp=drive_copy)

[Dilution\_fractions.csv](https://drive.google.com/open?id=1JGUnzcTTAi6hMbuh2tDD-FcpxbqoJxx1&usp=drive_copy)

[Initial\_DOC\_Values.csv](https://drive.google.com/open?id=1SoFvxgtjyeqqCdAAjSEd74HogNNqdQNQ&usp=drive_copy)

[Production\_Values.csv](https://drive.google.com/open?id=14Dr89eDtCHnkr61wR5A-uLxwuVD4lQub&usp=drive_copy)

[Tau\_values.csv](https://drive.google.com/open?id=1ZSi2diNgDxTtqplbr8Hi3DrExtzfGifQ&usp=drive_copy)

[Velocity\_Values.csv](https://drive.google.com/open?id=1Vc1BbRt_DLGwmZiBQO4QrXMKPSjLuKn2&usp=drive_copy) , which are saved as CSV files in the output/random\_initial\_val/ directory ([random\_initial\_val](https://drive.google.com/open?id=1Dm2LEWdGxyU8KJuFnaA1zm_YHySKtGdD&usp=drive_copy)).

This also generates **distribution plots of each input parameter**.

[Teslin\_river.py](https://drive.google.com/open?id=1yZtmOod1TNqkFkuEH37wTpxt1AmiQutp&usp=drive_copy)

This is the Teslin(first) tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/Teslin/ directory([teslin](https://drive.google.com/open?id=1klacGE0QGnZlPzFaFFloTXzwdm9EzAih&usp=drive_copy)).

[Teslin\_river](https://drive.google.com/open?id=1CpVc2hTtx33MS7ZOllsMNlp_GBg7gGbn&usp=drive_copy) [copy.py](http://copy.py)

A copy of the “[Teslin\_river.py](https://drive.google.com/open?id=1yZtmOod1TNqkFkuEH37wTpxt1AmiQutp&usp=drive_copy)” Python script

[Teslin\_river\_plots.py](https://drive.google.com/open?id=1V9XKN_NNd5U63SRhUtVxWWUy8cw8IiRY&usp=drive_copy)

This code is similar to the “[Teslin\_river.py](https://drive.google.com/open?id=1yZtmOod1TNqkFkuEH37wTpxt1AmiQutp&usp=drive_copy)”, but it contains an additional code part that generates plots of Chemical concentration vs distance.

[Koyukuk\_river.py](https://drive.google.com/open?id=1kiIgVnEwwU1ghGpLFlsyO49MK_rAhkaY&usp=drive_copy)

This is the Koyukuk tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/Koyukuk/ directory ([koyukuk](https://drive.google.com/open?id=1pAD2YLfUoc0xTJMDK8G8Ha5wMKTD3C2P&usp=drive_copy))

[Pelly\_river.py](https://drive.google.com/open?id=15HsP-hc6XRqMiBCfpwQqmJgDOE-_451n&usp=drive_copy)

This is the Pelly tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/Pelly/ directory ([pelly](https://drive.google.com/open?id=1Ra3cSI4s0t1xR6fuheNtgKivDy-PKR8Q&usp=drive_copy))

[Porcupine\_river.py](https://drive.google.com/open?id=1-OSNoNLG12lRJo62Pv9lKhNCmYFPomb7&usp=drive_copy)

This is the Porcupine tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/Porcupine/ directory ([porcupine](https://drive.google.com/open?id=1TJvoK7JuJujnUUrnUW01sAQ5aCb6R5w4&usp=drive_copy))

[Stewart\_river.py](https://drive.google.com/open?id=1uN2ttk1iKRAaSLF5C2dT8wOVqtnKKH7x&usp=drive_copy)

This is the Stewart tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/Stewart/ directory ([stewart](https://drive.google.com/open?id=1rQiTqlf9RXNgyu-MTKQGYqmeB7-J5pQt&usp=drive_copy))

[Tanana\_river.py](https://drive.google.com/open?id=1sq-OmMf7DRo1E9aJxVdEVOBvQPluRjJH&usp=drive_copy)

This is the Tanana tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/Tanana/ directory ([tanana](https://drive.google.com/open?id=18hAkW-uXODzh79xkcndLdQLKVFXp2dU4&usp=drive_copy))

[White\_Donjec\_river.py](https://drive.google.com/open?id=12XpmUGZWS8ZsQ-gNQe0Ys9qh-XgmXYfP&usp=drive_copy)

This is the White\_Donjec tributary module of the Yukon River. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/White\_Donjec/ directory ([white+donjec](https://drive.google.com/open?id=1OO5jz7pSOOUK0Fzd45SOeGE_XjG7pl33&usp=drive_copy))

[Yukon\_river.py](https://drive.google.com/open?id=1gwh1bZR3DvVEJkI_GKR9tk5js1Ggqi_F&usp=drive_copy)

This is the main module of the Yukon River, which represents the Yukon River stem. This is the script that calls all the tributary modules to the main river code in this script. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

But since this is the main river code of the stem, it contains the dilution function at the node points where tributaries are connecting to the main stem.

*Output of the code*

The output of this script is the river mouth concentration of the tributary, which is saved as a CSV file in the output/White\_Donjec/ directory ([yukon](https://drive.google.com/open?id=1fWe2rrwjifOMPZq7bDnswTYYqGr7fHW4&usp=drive_copy))

[Yukon\_river\_plot.py](https://drive.google.com/open?id=1KpQeUl_bpUXS2IKa5E2-9J2hjWxGJbHI&usp=drive_copy)

This is the main module of the Yukon River, which represents the Yukon River stem. It contains Python code that includes:

* A function for chemical decay,
* A loop for Monte Carlo analysis,
* Code to generate values for imax (maximum number of iterations), tmax (maximum number of time steps), and dmax (maximum distance of the tributary),
* Time step calculations (time\_calculation),
* Distance calculations (distance\_calculation),
* Creation of arrays for C\_Conc, Tau, and Prod,
* Initialization of C\_Conc, Tau, and Prod values for the chemical
* The main river code
* Creating CSV files that have the tributary river mouth concentrations of each macromolecule

*Output of the code*

In addition to the Yukon\_river.py script, this Python code generates a CSV file that stores output data from all iterations across all time steps for n sample runs (e.g., from a Monte Carlo simulation).

For example, in a single sample run, the concentration is recorded at each time step of the simulation. This data can then be used to plot distance vs. concentration, illustrating how concentration changes along the length of the tributary or river over time.

[Data\_Analysis\_1.py](https://drive.google.com/open?id=16Qk64IHK3Nr-cBctbnByVTAhr_BxQyYd&usp=drive_copy)

This code generates plots of DOC concentration versus distance along the Yukon River main stem. The plot also includes observational data of DOC concentrations at corresponding distances along the river course.

*Output of the code*

Plots of DOC concentration versus distance along the Yukon River main stem

[Data\_Ana\_2.py](https://drive.google.com/open?id=1T7XOzMq7C8FlI4VffPiD4-uOxE5bRY7a&usp=drive_copy)

This is the main (ultimate) code. Once you set the initial average values and parameter boundaries in River\_initial\_monac\_code.py, there is no need to manually run the separate tributary or Yukon\_river.py scripts.

*Output of the code*

The code automatically calls the necessary modules and generates the following:

* **Plots of DOC concentration vs. input variables**

This code also generates an analysis folder, located at /output/analysis/, which combines all the CSV files needed for further analysis. ([analysis](https://drive.google.com/open?id=1He76INuI5yqH-0WZWyHgOAkQjcv4WO5H&usp=drive_copy))

[CDOM\_component\_fractions.csv](https://drive.google.com/open?id=1_fll54B4NXKvilW5erfF8506JZe775ph&usp=drive_copy)

[Chemical\_fraction.csv](https://drive.google.com/open?id=1MP_4mDO7fcvEsA_OBJ-XeF9v9ZOe4H2B&usp=drive_copy)

[Dilution\_fractions.csv](https://drive.google.com/open?id=1BB8hSNwvE8lEkyE1ugpHZcZofLS-06Aa&usp=drive_copy)

[Initial\_DOC\_Values.csv](https://drive.google.com/open?id=1prRj9PejQgiciAPv1GJdAeXtFYW0_t38&usp=drive_copy)

[InputData.csv](https://drive.google.com/open?id=1xV8cU0dRYYp-YNY5mvm1bbsRkKAmLTO5&usp=drive_copy)

[Koyukuk\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=1325CO4c0JyY-jMDOv1UhnywbP9_uozBE&usp=drive_copy)

[OutputData.csv](https://drive.google.com/open?id=1P5Vw_TodVF9Wr7mTaAxdTeJW62ihQkLs&usp=drive_copy)

[Pelly\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=1ty_4uRZNelOuRhLKDuT-0uZpH7dajCEx&usp=drive_copy)

[Porcupine\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=16Lr6iCjoHDIItMSE9gaIhl_PV34srMVs&usp=drive_copy)

[Production\_Values.csv](https://drive.google.com/open?id=1RntfM1Zj5f11AVf9L_15r5VuXGlOjvdT&usp=drive_copy)

[Stewart\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=1oLaBuqziD4-orZgZE3fEWdiJsBIUPkSk&usp=drive_copy)

[Tanana\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=1iQqUesd1OwDRYAE-IED1aRCoIP-7Ddk1&usp=drive_copy)

[Tau\_Values.csv](https://drive.google.com/open?id=1P-yn1Cp9L-tfkd2SWhX8KuF7gSrW3yFF&usp=drive_copy)

[Teslin\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=10HHVhrJaC7fHEKCiehYUVAhNDbuPgLSJ&usp=drive_copy)

[Velocity\_Values.csv](https://drive.google.com/open?id=1iQtZUN8caTnHyADlDRPET2e-a2dK7NOX&usp=drive_copy)

[White+Donjec\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=1Megg3akxUI6bVRCJtUlXlNZ-2Zxi5KHM&usp=drive_copy)

[Yukon\_River\_Mouth\_Values.csv](https://drive.google.com/open?id=18JPYWii6P9LerxD5TpYC-MY42Q5pRBxv&usp=drive_copy)

Testing python codes

[Test\_0.py](https://drive.google.com/open?id=1Ww10YG9C3dNUEq3kz7k3hWkYEuPpjwZp&usp=drive_copy)

[Just\_in\_case.py](https://drive.google.com/open?id=1AyZtm4W--MzTIDoDDZrvr74GbOhVQDDd&usp=drive_copy)

[node\_headings.py](https://drive.google.com/open?id=1Rs6O2FszWCmdcvQIrEASTCQguFPeDIFN&usp=drive_copy)