Monte Python: analyzing runs with basic options

Benjamin Audren

École Polytechnique Fédérale de Lausanne

28/10/2014

Main information

To keep in mind

- Since all the information is contained in the **output folder**, it is only necessary to specify this to analyze a run.
- Analyze **only chains produced with the same covariance matrix**, or **jumping factor**.
- The analysis produces **covariance matrix, best-fit files, and posterior information**.
- The covariance and best-fit information can be used in **future runs**.

Basic analysis

Commands

```
python montepython/MontePython.py info chains/jla
```

This analyses **all chains** in the folder, *regardless* of their possibly different covariance matrix.

Do it!

careful, it requires LaTeX

Basic analysis

```
 Running Monte Python v2.1.0

—> Finding global maximum of likelihood
—> Removing burn—in
—> Scanning file chains/jla_full/2014—10—07_100000__3.txt : Removed 152 points of burn—in
                                 2014—10—07_100000__4.txt : Removed 79  points of burn—in
                                 2014—10—07_100000__1.txt : Removed 50  points of burn—in
                                 2014—10—07_100000__2.txt : Removed 61  points of burn—in
—> Computing mean values
—> Computing variance
—> Computing convergence criterium (Gelman—Rubin)
 —> R is 0.180485       for  Omega_cdm
         0.127232       for  alpha
         0.042833       for  beta
         0.081718       for  M
         0.452856       for  Delta_M
         0.180535       for  Omega_m
         0.180485       for  Omega_Lambda
—> Computing covariance matrix
————————————————————————————————————————————————
 —> Computing histograms for  Omega_cdm
  /!\ could not derive minimum credible intervals for this multimodal posterior
 —> Computing histograms for  alpha
 —> Computing histograms for  beta
 —> Computing histograms for  M
 —> Computing histograms for  Delta_M
 —> Computing histograms for  Omega_m
 —> Computing histograms for  Omega_Lambda
————————————————————————————————————————————————
—> Saving figures to .pdf files
—> Writing .info and .tex files
```

Basic analysis

```
 Running Monte Python v2.1.0

--> Finding global maximum of likelihood
--> Removing burn-in
--> Scanning file chains/jla_full/2014-10-07_200000__9.txt   : Removed 0  points of burn-in
...
                                2014-10-07_200000__6.txt   : Removed 0  points of burn-in
--> Computing mean values
--> Computing variance
--> Computing convergence criterium (Gelman-Rubin)
 -> R is 0.000911       for  Omega_cdm
         0.000629       for  alpha
         0.000457       for  beta
         0.000575       for  M
         0.000679       for  Delta_M
         0.000911       for  Omega_m
         0.000911       for  Omega_Lambda
--> Computing covariance matrix
-----------------------------------------
 -> Computing histograms for  Omega_cdm
 -> Computing histograms for  alpha
 -> Computing histograms for  beta
 -> Computing histograms for  M
 -> Computing histograms for  Delta_M
 -> Computing histograms for  Omega_m
 -> Computing histograms for  Omega_Lambda
-----------------------------------------
--> Saving figures to .pdf files
--> Writing .info and .tex files
```

Basic analysis

output

- plots are located in `chains/jla/plots/`
- convergence information in `jla.log`, `jla.v_info`
- useful information for future runs: `jla.covmat`, `jla.bestfit`

Principle

Guidelines

- Depends on the **number** of parameters and degeneracy.
- If $0.1 \geq R$ for every parameter, **use** the new covmat, bestfit.
- **Copy the new covmat, besfit** to a safe place.
- Else, **make new chains** with the initial proposal density.

Principle

Guidelines

- Depends on the **number** of parameters and degeneracy.
- If $0.1 \geq R$ for every parameter, **use** the new covmat, bestfit.
- **Copy the new covmat, besfit** to a safe place.
- Else, **make new chains** with the initial proposal density.

Warning

Whatever happens, **do not** analyze chains with different proposal densities.

What to do when it does not work

Low acceptance rate

Emergency procedure

- **Situation**: **very low** acceptance rate, very few points.
- **Cause**: **steps too large**
- **What to do**: **use the `-f x` flag** to reduce the step size

What to do when it does not work

Low acceptance rate

Emergency procedure

- **Situation**: **very low** acceptance rate, very few points.
- **Cause**: **steps too large**
- **What to do**: **use the `-f x` flag** to reduce the step size

the -f flag

- Had-oc parameter, default to **2.4** for N-dimensional gaussians
- Is there to fix the acceptance rate to good values (0.2, 0.3)
- Nothing prevents you from **changing it** ...
- But remember to **not combine chains with different f**
- **How?**: use different chain numbers!

What to do when it does not work

High acceptance rate

Emergency procedure

- **Situation**: **very high** acceptance rate, points too correlated.
- **Cause**: **steps too small**, no exploration
- **What to do**: **use the `-f x` flag** to augment the step size

I tried everything, and it still does not work

Metropolis-Hastings weaknesses

- MH is bad in **highly non-gaussian** cases
- It is possible to help him with **guessing well enough** the step sizes (editing the parameter file)
- Remember to use **Cholesky decomposition** with **large number of nuisance parameters**.
- If nothing works, use **EMCEE** (Thursday)

Options when analyzing

there are many options

- Try `python montepython/MontePython.py info --help`
- You can **customize most of the things** (font size, colors, number of ticks, legend)
- Compare several folders (`info folder_1 folder_2`)
- Default colors and colormaps are **grey-scale friendly**

Main options

- Remove the mean-likelihood with `--no-mean`
- Only compute the covmat with `--noplot`
- Output all sub plots with `--all`

Using an extra file for plotting options

Alternative to command line

- example in `plot_files/example.plot`
- syntax: `--extra plot_files/example.plot`

Using an extra file for plotting options

Alternative to command line

- example in `plot_files/example.plot`
- syntax: `--extra plot_files/example.plot`

4 exclusive arguments

- **redefine**: dict for **combining parameters** (tomorrow)
- **to change**: dict for **renaming**
- **new scales**: dict for **rescaling**
- **to plot**: list for **plotting new names**

Using an extra file for plotting options

Alternative to command line

- example in `plot_files/example.plot`
- syntax: `--extra plot_files/example.plot`

4 exclusive arguments

- **redefine**: dict for **combining parameters** (tomorrow)
- **to change**: dict for **renaming**
- **new scales**: dict for **rescaling**
- **to plot**: list for **plotting new names**

More options

Tomorrow!

Example

Reducing the number of plotted parameters

```
info.to_plot = ['Omega_cdm', 'alpha']
```

Exercise

plot only the nuisance parameter 2d posterior information.

Renaming parameters

```
info.to_change  = {'Omega_cdm': r'$\Omega_{\rm cdm}$', 'alpha': '$\gamma$'}
info.new_scales = {'$\gamma$': 10}
info.to_plot    = [r'$\Omega_{\rm cdm}$', '$\gamma$', 'beta']
```