Taller de Cosmología con SimpleMC 2.0.0

Día 1: Agregando un modelo

3-5 de junio de 2020

Contents

La clase DriverMC

La clase DriverMC

- Es la clase principal, conecta todos los modulos y clases de SimpleMC.
- Se le puede dar como argumento un `inifile.ini` con toda la configuración deseada, o bien, escribir los parámetros de la función con los valores deseados.
- Si se omite un valor, se usarán valores por default (salvo en el modelo).

.

DriverMC con **kwargs

```python
from simplemc.DriverMC import DriverMC
# DriverMC class con el modelo, datos y carpeta de outputs.
analyzer = DriverMC(model=LCDM, analyzername="mcmc", datasets='HD+BBAO', chainsdir="chains")
# Características del analyzer elegido
analyzer.executer(nsamp=1000, skip=0)
analyzer.postprocess()
fig = analyzer.plot(show=True)
fig.simpleGetdist(smooth2d=0.1, smooth1d=0.1)
```

DriverMC con iniFile=archivo.ini

```python
from simplemc.DriverMC import DriverMC
# Usar un ini file y cargarlo en DriverMC vía iniFile
inifile = "baseConfig.ini"
# DriverMC class con el modelo, datos y carpeta de outputs.
analyzer = DriverMC(iniFile=inifile)
# Características del analyzer elegido
analyzer.executer()
analyzer.postprocess()
fig = analyzer.plot(show=True)
fig.simpleGetdist(smooth2d=0.1, smooth1d=0.1)
```

Ejemplo de ini file

```
[custom]
;directory for chains/output
chainsdir = chains
;set model
;model options: LCDM, LCDMasslessnu, nuLCDM, NeffLCDM, noradLCDM, nuoLCDM,
;nuwLCDM, oLCDM, wCDM, waCDM, owCDM, owaCDM, JordiCDM, WeirdCDM, TLight, StepCDM,
;Spline, PolyCDM, fPolyCDM, Decay, Decay01, Decay05, EarlyDE, EarlyDE_rd_DE, SlowRDE, sline
;more options located in the RunBase.py
model = simpleCosmo
;varys8 True otherwise s8=0.8
varys8  = False
;set datasets used. Ex: UnionSN+BBAO+Planck
;data options: HD, BBAO, GBAO, GBAO_no6dF, CMASS, LBAO, LaBAO, LxBAO, MGS, Planck, WMAP, PlRd, WRd, PlDa, PlRdx10, CMBW, SN, SNx10,
UnionSN, RiessH0, 6dFGS, dline
datasets = HD+SN
;sampler can be {mcmc, nested, emcee}
;or analyzers {maxlike, genetic}
;
;mcmc -> metropolis-hastings
;nested
;emcee
;maxlike -> Maximum Likelihood Analyzer
;genetic
analyzername = mcmc
[mcmc]
;Nsamples
nsamp   = 5000
;Burn-in
skip    = 0
;if single cpu, otherwise use mpi -np #
chainno = 1
```

Agregar un modelo no cosmológico

Ejemplo 1: testModel.py

```
from simplemc.models.SimpleModel import SimpleModel
from simplemc.cosmo.Parameter import Parameter
from simplemc.DriverMC import DriverMC
```

Paréntesis: La clase Parameter

```
class Parameter:
    def __init__(self, name, value, err=0.0, bounds=None, Ltxname=None):
        self.name = name
        if Ltxname:
            self.Ltxname = Ltxname
        else:
            self.Ltxname = name
        self.value = value
        # this is the estimate of error
        self.error = err
        if bounds == None:
            self.bounds = (value-5*err, value+5*err)
        else:
            self.bounds = bounds
```

Volver a testModel.py y correrlo

```
# 1) Define your parameters objects
# name string,  value intermediate, step size,
# (bound_inf, bound_sup),  LaTeX name
m = Parameter("m", 0.5, 0.01, (0,5), "\m_0")
b = Parameter("b", 0.5, 0.01, (0,5), "\b_0")

# 2) Create a list with your parameters objects
parameterlist = [m, b]

# 3) Define a method that reads a list of parameters,
# unzip them and return the a function of x with the
# parameters.
def model(parameterlist, x):
    m, b = parameterlist
    return m*x+b+10
```

Agregar un modelo cosmológico

Clases LCDMCosmology y BaseCosmology

- BaseCosmology es el esqueleto de un modelo cosmológico.
- LCDMCosmology (como su nombre lo indica) es el modelo cosmológico estándar.

Clase LCDMCosmology como base de otros modelos

- Los demás modelos en SimpleMC son extensiones de la clase LCDMCosmology.

```
class owa0CDMCosmology(LCDMCosmology):
    def __init__(self, varyw=True, varywa=True, varyOk=True):
        # three parameters: w, wa, Ok

        self.varyw  = varyw
        self.varywa = varywa
        self.varyOk = varyOk

        self.Ok = Ok_par.value
        self.w0 = w_par.value
        self.wa = wa_par.value
        LCDMCosmology.__init__(self)
```

```
class QuintomCosmology(LCDMCosmology):
    def __init__(self, varymquin=False,
        ## two parameters: Om and h

        """Is better to start the chai
        may take much longer"""
```

Modificando la clase simplemc.models.SimpleCosmoModel

```python
from .LCDMCosmology import LCDMCosmology
from simplemc.cosmo.Parameter import Parameter
from simplemc.cosmo.paramDefs import Ok_par, w_par, wa_par
import math as N

class SimpleCosmoModel(LCDMCosmology):
    def __init__(self):
        # Create a list with your parameters,
        # pull it from paramDefs module or create them
        # with the Parameter class
        # self.nuevoparm = Parameter("nombre", 0.5)
        self.parameters = [w_par]
        # .value for each new parameter to a new variable class.
        self.w0 = w_par.value

        LCDMCosmology.__init__(self)
    # my free parameters. We add Ok on top of LCDM ones (we inherit LCDM)
    def freeParameters(self):
        l = LCDMCosmology.freeParameters(self)
        for parameter in self.parameters:
            l.append(parameter)
        return l

    def updateParams(self, pars):
        ok = LCDMCosmology.updateParams(self, pars)
        if not ok:
            return False
        return True
```

Paréntesis: el módulo simplemc.runbase

Dados las cadenas de texto para los modelos y datos, instancia los objetos correspondientes de modelos y likelihoods

- ParseModel
- ParseDataset

Recomendación:
Al agregar un nuevo modelo cosmológico via SimpleCosmoModel, se evita modificar el módulo runbase.

Analizar modelo vía un test.py

```
analyzer = DriverMC(model="simpleCosmo", analyzername="nested", chainsdir="chains")
analyzer.executer(nlivepoints=50)
analyzer.postprocess()
fig = analyzer.plot(show=True)
fig.simpleGetdist(smooth2d=0.5, smooth1d=0.5)
```

- Correr script.
- Revisar salidas de texto y figuras.
- Comparar salidas com las de LCDM.
- Comentar algunas opciones para graficar.

Mañana

Mañana

- Añadir datos
- Comparar modelos

Bibliography

- Foreman-Mackey, D. (2017). corner. py: Corner plots. Astrophysics Source Code Library.
- Lewis, A. (2019). GetDist: a Python package for analysing Monte Carlo samples. arXiv preprint arXiv:1910.13970.