

Third task

- Download the SPT fgas data
 - Fit the data to $f_0(1 + f_1z)$ where f_0 and f_1 are unknown constants
 - Determine the best fit values of f_0 and f_1 including 68% and 90% credible intervals using emcee and corner.py
 - The priors on f_0 and f_1 should be $0 < f_0 < 0.5$ and $-0.5 < f_1 < 0.5$.
 - Use the same likelihood as in equation 6 of <https://arxiv.org/pdf/2001.08340.pdf> (radial acceleration relation for galaxy cluster
 - download emcee and python corner module and look up <https://emcee.readthedocs.io/en/stable/tutorials/line/> which shows how to fit a model to a straight line
- ```
import numpy as np
```

```
import scipy as sp

In [2]: fname = np.loadtxt('fgas_spt.txt', delimiter=" ") #loading the given data
print('the structure is Z, fgas, fgas_error, ignore', fname[0]) #checking for the structure of the element
```

```
the structure is z, fgas, fgas_error, ignore [0.2777 0.09661017 0.01488267 0.]
```

```
In [3]: # arrays as given in the data
```

```
fgas = [] #hot gas fraction
fgas_err = [] #corresponding error in the observations
ignore = []


for i in fname:
 z.append(i[0])
 fgas.append(i[1])
 fgas_err.append(i[2])
 ignore.append(i[3])

x = np.array(z)
y = np.array(fgas)
yerr = np.array(fgas_err)
```

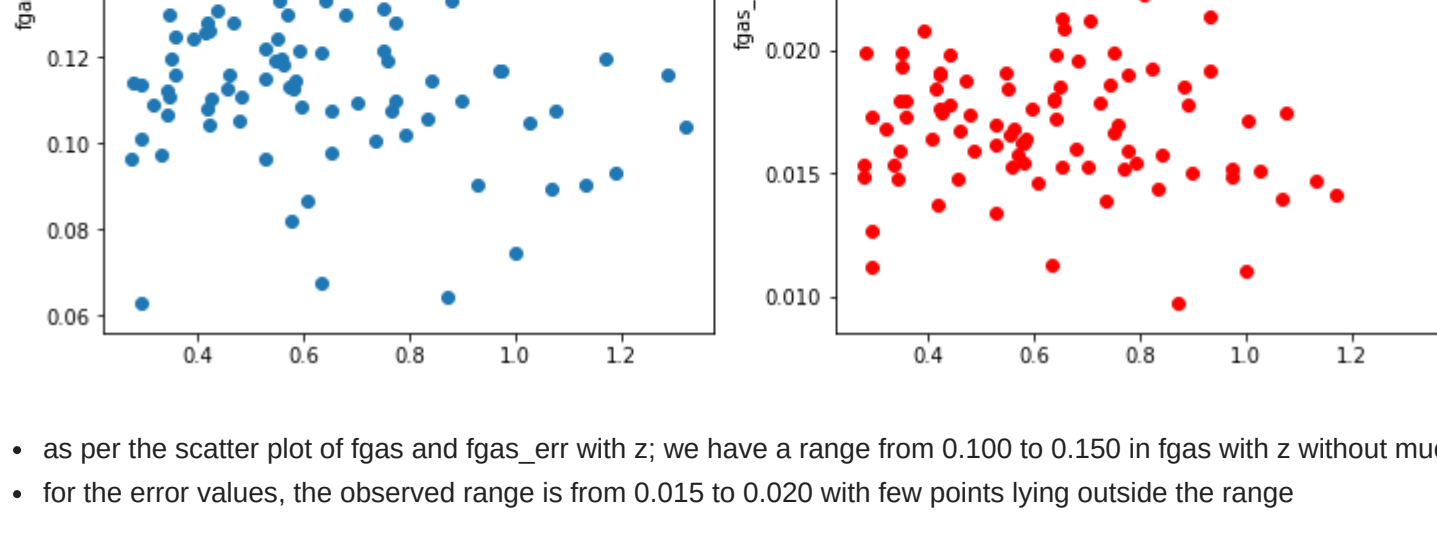
In [4]:

```
fig, ax = plt.subplots(1,2, figsize=(12,6))
ax[0].scatter(x, y)
ax[1].scatter(x,yerr, color='r')
ax[0].set_ylabel('fgas')
ax[1].set_ylabel('fgas_err')
```

plt.show()



The figure displays two scatter plots side-by-side. The left plot shows the relationship between 'x' (ranging from 0 to 100) and 'tr' (ranging from 0.14 to 0.20). The data points are blue dots, showing a general downward trend with some fluctuations. The right plot shows the relationship between 'x' (ranging from 0 to 100) and 'tr' (ranging from 0.025 to 0.035). The data points are red dots, showing a clear upward trend.



- the function used is a log-likelihood function given as

$$\ln \mathcal{L} =$$

data points and  $\sigma_i$  includes the observational uncertainty

- $\sigma_{int}$  accounts for the intrinsic scatter around the mean RAR (radial acceleration relation) due to unaccounted astrophysics associated with the RAR
- ```
#coding the likelihood function
def log_likelihood(theta, x, y, yerr):
    m = theta[0]
    b = theta[1]
```

```
log_r = theta[2]

model = m*x + b
sigma2 = yerr**2 + model**2 * np.exp(2 * log_f)
return -0.5 * np.sum((y - model) ** 2 / sigma2 + np.log(2*np.pi*sigma2))
```

- as per the standard equation and the given model $f_0 * (1 + f_1 * z)$; we have $m = f_0 * f_1$ and $b = f_0 \implies f_0 = b \ \& \ f_1 = \frac{m}{f_0}$

with

```
f_true = np.std(ignore) #fractional amount of underestimation

nll = lambda *args: -log_likelihood(*args)
initial = np.array([m_true, b_true, np.log(f_true)]) #making array of true params
soln = minimize(nll, initial, args=(x, y, yerr)) #minimized the -ve likelihood function
m_ml, b_ml, log_f_ml = soln.x #output of the values

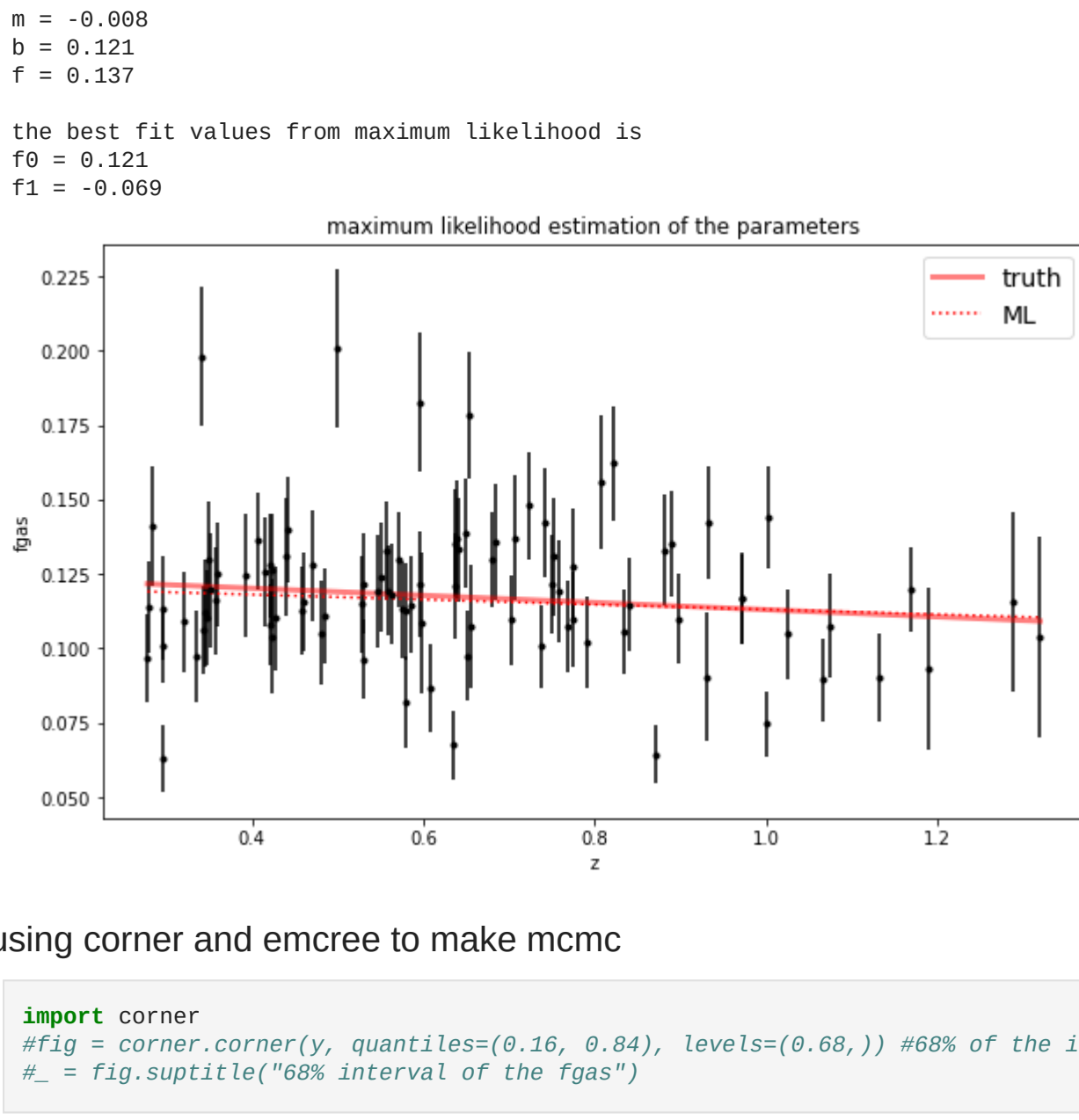
f0 = b_ml
f1 = m_ml/b_ml

print("Maximum likelihood estimates:")
print("m = {0:.3f}".format(m_ml))
print("b = {0:.3f}".format(b_ml))
print("f = {0:.3f} \n".format(np.exp(log_f_ml)))

print('the best fit values from maximum likelihood is')
print("f0 = {0:.3f}".format(b_ml))
print("f1 = {0:.3f}".format(m_ml/b_ml))

plt.figure(figsize=(10,6))
plt.errorbar(x, y, yerr=yerr, fmt="k", capsize=0)
plt.plot(x, m_true * x + b_true, "r", alpha=0.5, lw=3, label="truth") #true linear plot
#plt.plot(x, np.dot(np.vander(x, 2), w), "--k", label="LS")
plt.plot(x, np.dot(np.vander(x, 2), [m_ml, b_ml]), "r", label="ML") #maximum likelihood function estimation
plt.legend(fontsize=14)

#plt.xlim(0, 10)
plt.xlabel("z")
plt.ylabel('fgas')
plt.title('maximum likelihood estimation of the parameters')
plt.show()
```



```
#making a prior function
```

```
def log_prior(theta):
    m, b, log_f = theta
    if -0.25 < m < 0.25 and 0.0 < b < 0.5 and -5.0 < log_f < 1.0:
        return 0.02 #not sure what is this
    return -np.inf
```

```
In [78]: #log_probability function
def log_probability(theta, x, y, verr):
```

```
lp = log_prior(theta)
if not np.isfinite(lp):
    return -np.inf
return lp + log_likelihood(theta, x, y, yerr)
```

```
In [79]: import emcee

pos = soln.x + 1e-4 * np.random.randn(32, 3)
nwalkers, ndim = pos.shape

sampler = emcee.EnsembleSampler(
    nwalkers, ndim, log_probability, args=(x, y, yerr)
)
sampler.run_mcmc(pos, 5000, progress=True);
```

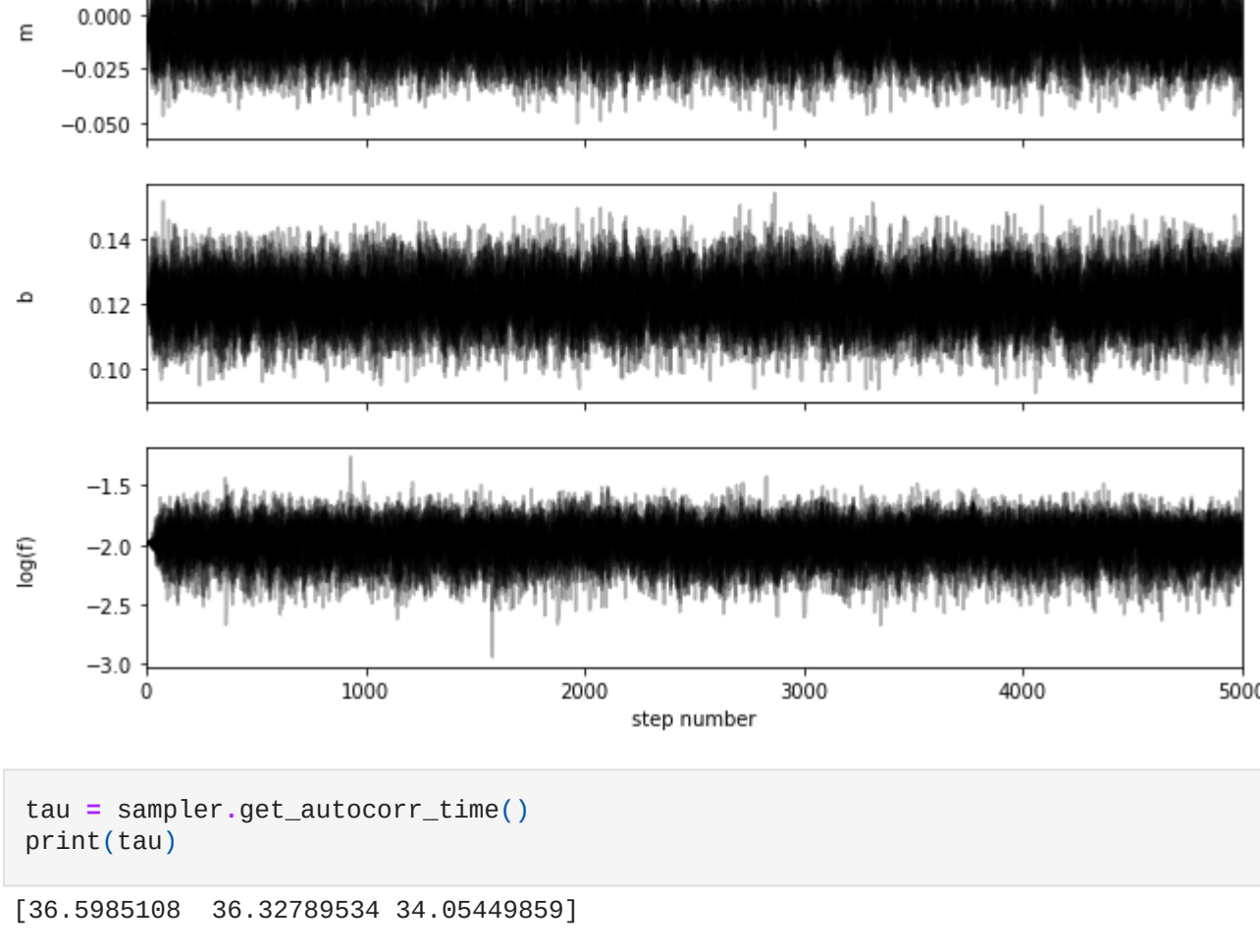
```
In [80]: fig, axes = plt.subplots(3, figsize=(10, 7), sharex=True)
         samples = sampler.get_chain()
```

```
for i in range(ndim):
    ax = axes[i]
    ax.plot(samples[:, :, i], "k", alpha=0.3)
    ax.set_xlim(0, len(samples))
    ax.set_ylabel(labels[i])
    ax.yaxis.set_label_coords(-0.1, 0.5)

axes[-1].set_xlabel("step number");
```



The plot displays a highly volatile time series. The y-axis has a maximum tick at 0.025. The x-axis represents the progression of steps. The data is plotted as a dense, dark line with many sharp peaks and troughs, indicating significant noise or high-frequency fluctuations in the signal.



```
In [82]: flat_samples = sampler.get_chain(discard=100, thin=15, flat=True)
         print(flat_samples.shape)
```

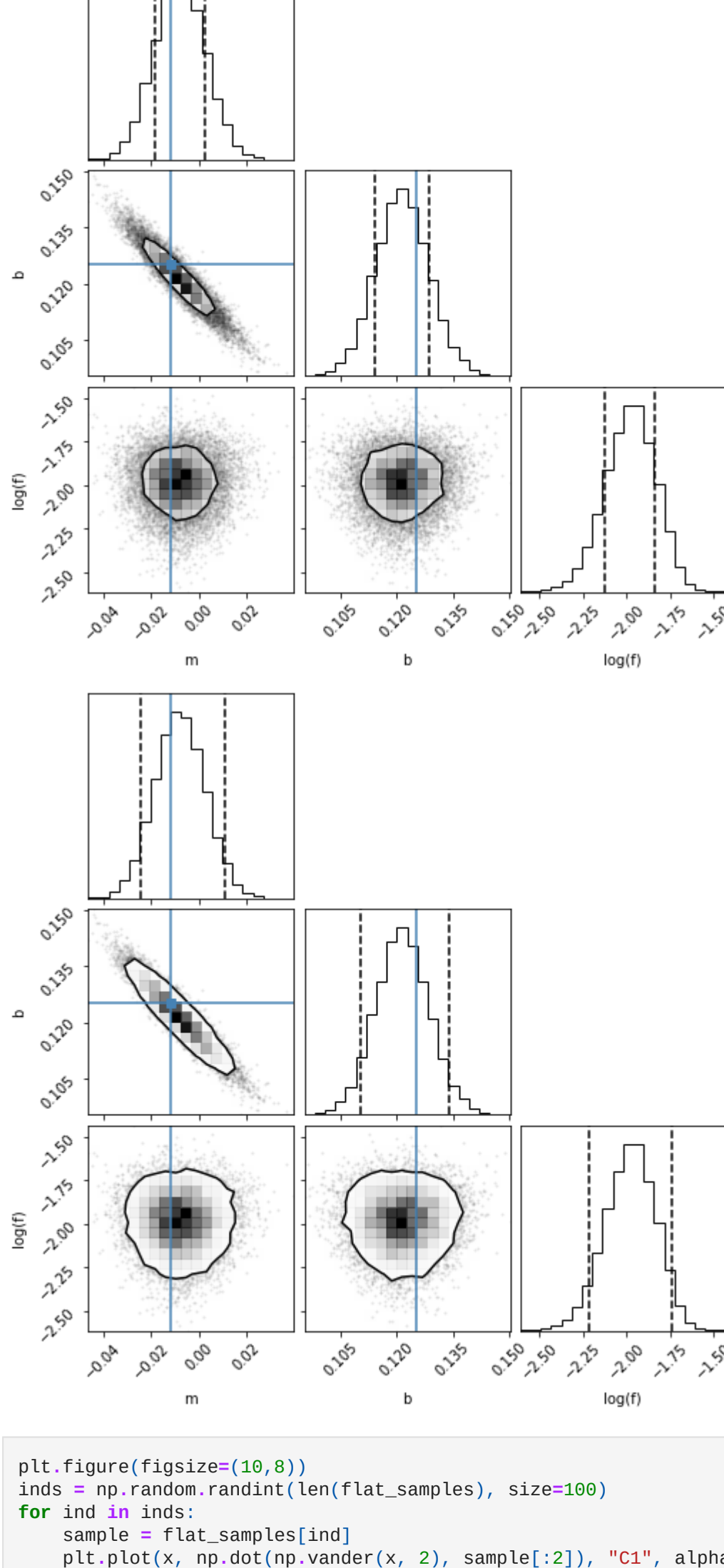
```
generating corner plots
```

```
In [90]: import corner

fig = corner.corner(
    flat_samples, labels=labels, truths=[m_true, b_true, np.log(f_true)], levels=(0.68, ), quantiles=(0.16, 0.84, )
```

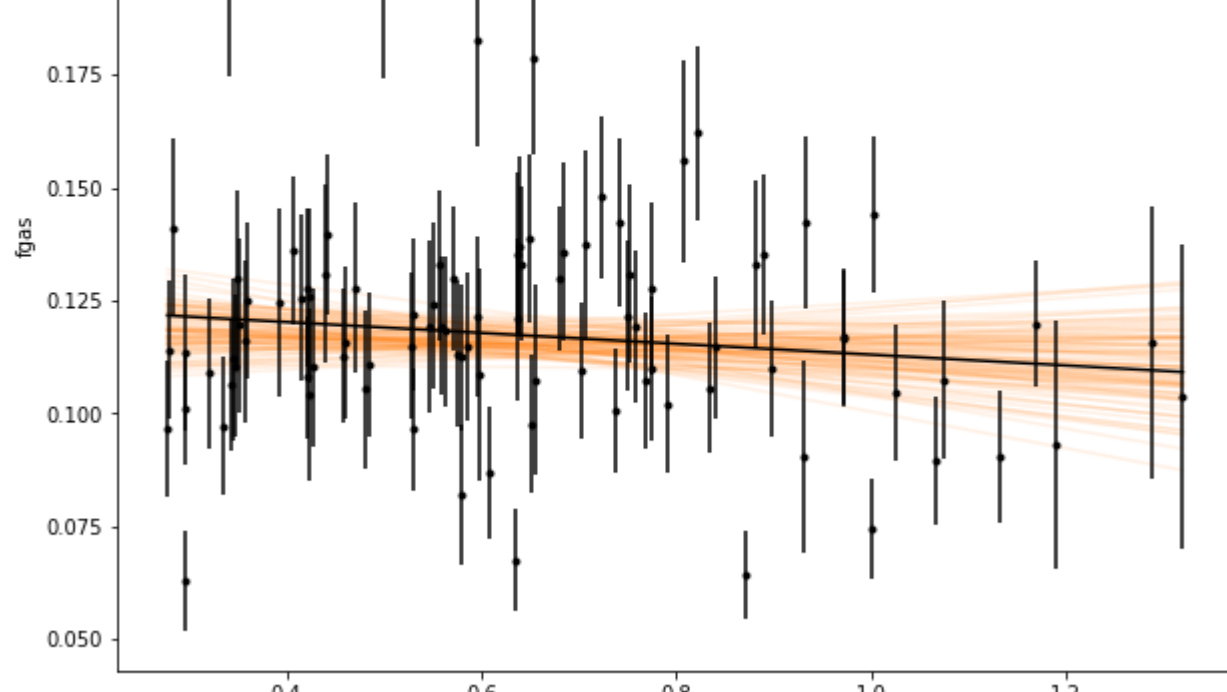
```
);  
  
fig = corner_corner(  
    x = x, y = y, z = z, w = w,  
    x2 = x2, y2 = y2, z2 = z2, w2 = w2,  
    x3 = x3, y3 = y3, z3 = z3, w3 = w3,  
    x4 = x4, y4 = y4, z4 = z4, w4 = w4,  
    x5 = x5, y5 = y5, z5 = z5, w5 = w5,  
    x6 = x6, y6 = y6, z6 = z6, w6 = w6,  
    x7 = x7, y7 = y7, z7 = z7, w7 = w7,  
    x8 = x8, y8 = y8, z8 = z8, w8 = w8,  
    x9 = x9, y9 = y9, z9 = z9, w9 = w9,  
    x10 = x10, y10 = y10, z10 = z10, w10 = w10,  
    x11 = x11, y11 = y11, z11 = z11, w11 = w11,  
    x12 = x12, y12 = y12, z12 = z12, w12 = w12,  
    x13 = x13, y13 = y13, z13 = z13, w13 = w13,  
    x14 = x14, y14 = y14, z14 = z14, w14 = w14,  
    x15 = x15, y15 = y15, z15 = z15, w15 = w15,  
    x16 = x16, y16 = y16, z16 = z16, w16 = w16,  
    x17 = x17, y17 = y17, z17 = z17, w17 = w17,  
    x18 = x18, y18 = y18, z18 = z18, w18 = w18,  
    x19 = x19, y19 = y19, z19 = z19, w19 = w19,  
    x20 = x20, y20 = y20, z20 = z20, w20 = w20,  
    x21 = x21, y21 = y21, z21 = z21, w21 = w21,  
    x22 = x22, y22 = y22, z22 = z22, w22 = w22,  
    x23 = x23, y23 = y23, z23 = z23, w23 = w23,  
    x24 = x24, y24 = y24, z24 = z24, w24 = w24,  
    x25 = x25, y25 = y25, z25 = z25, w25 = w25,  
    x26 = x26, y26 = y26, z26 = z26, w26 = w26,  
    x27 = x27, y27 = y27, z27 = z27, w27 = w27,  
    x28 = x28, y28 = y28, z28 = z28, w28 = w28,  
    x29 = x29, y29 = y29, z29 = z29, w29 = w29,  
    x30 = x30, y30 = y30, z30 = z30, w30 = w30,  
    x31 = x31, y31 = y31, z31 = z31, w31 = w31,  
    x32 = x32, y32 = y32, z32 = z32, w32 = w32,  
    x33 = x33, y33 = y33, z33 = z33, w33 = w33,  
    x34 = x34, y34 = y34, z34 = z34, w34 = w34,  
    x35 = x35, y35 = y35, z35 = z35, w35 = w35,  
    x36 = x36, y36 = y36, z36 = z36, w36 = w36,  
    x37 = x37, y37 = y37, z37 = z37, w37 = w37,  
    x38 = x38, y38 = y38, z38 = z38, w38 = w38,  
    x39 = x39, y39 = y39, z39 = z39, w39 = w39,  
    x40 = x40, y40 = y40, z40 = z40, w40 = w40,  
    x41 = x41, y41 = y41, z41 = z41, w41 = w41,  
    x42 = x42, y42 = y42, z42 = z42, w42 = w42,  
    x43 = x43, y43 = y43, z43 = z43, w43 = w43,  
    x44 = x44, y44 = y44, z44 = z44, w44 = w44,  
    x45 = x45, y45 = y45, z45 = z45, w45 = w45,  
    x46 = x46, y46 = y46, z46 = z46, w46 = w46,  
    x47 = x47, y47 = y47, z47 = z47, w47 = w47,  
    x48 = x48, y48 = y48, z48 = z48, w48 = w48,  
    x49 = x49, y49 = y49, z49 = z49, w49 = w49,  
    x50 = x50, y50 = y50, z50 = z50, w50 = w50,  
    x51 = x51, y51 = y51, z51 = z51, w51 = w51,  
    x52 = x52, y52 = y52, z52 = z52, w52 = w52,  
    x53 = x53, y53 = y53, z53 = z53, w53 = w53,  
    x54 = x54, y54 = y54, z54 = z54, w54 = w54,  
    x55 = x55, y55 = y55, z55 = z55, w55 = w55,  
    x56 = x56, y56 = y56, z56 = z56, w56 = w56,  
    x57 = x57, y57 = y57, z57 = z57, w57 = w57,  
    x58 = x58, y58 = y58, z58 = z58, w58 = w58,  
    x59 = x59, y59 = y59, z59 = z59, w59 = w59,  
    x60 = x60, y60 = y60, z60 = z60, w60 = w60,  
    x61 = x61, y61 = y61, z61 = z61, w61 = w61,  
    x62 = x62, y62 = y62, z62 = z62, w62 = w62,  
    x63 = x63, y63 = y63, z63 = z63, w63 = w63,  
    x64 = x64, y64 = y64, z64 = z64, w64 = w64,  
    x65 = x65, y65 = y65, z65 = z65, w65 = w65,  
    x66 = x66, y66 = y66, z66 = z66, w66 = w66,  
    x67 = x67, y67 = y67, z67 = z67, w67 = w67,  
    x68 = x68, y68 = y68, z68 = z68, w68 = w68,  
    x69 = x69, y69 = y69, z69 = z69, w69 = w69,  
    x70 = x70, y70 = y70, z70 = z70, w70 = w70,  
    x71 = x71, y71 = y71, z71 = z71, w71 = w71,  
    x72 = x72, y72 = y72, z72 = z72, w72 = w72,  
    x73 = x73, y73 = y73, z73 = z73, w73 = w73,  
    x74 = x74, y74 = y74, z74 = z74, w74 = w74,  
    x75 = x75, y75 = y75, z75 = z75, w75 = w75,  
    x76 = x76, y76 = y76, z76 = z76, w76 = w76,  
    x77 = x77, y77 = y77, z77 = z77, w77 = w77,  
    x78 = x78, y78 = y78, z78 = z78, w78 = w78,  
    x79 = x79, y79 = y79, z79 = z79, w79 = w79,  
    x80 = x80, y80 = y80, z80 = z80, w80 = w80,  
    x81 = x81, y81 = y81, z81 = z81, w81 = w81,  
    x82 = x82, y82 = y82, z82 = z82, w82 = w82,  
    x83 = x83, y83 = y83, z83 = z83, w83 = w83,  
    x84 = x84, y84 = y84, z84 = z84, w84 = w84,  
    x85 = x85, y85 = y85, z85 = z85, w85 = w85,  
    x86 = x86, y86 = y86, z86 = z86, w86 = w86,  
    x87 = x87, y87 = y87, z87 = z87, w87 = w87,  
    x88 = x88, y88 = y88, z88 = z88, w88 = w88,  
    x89 = x89, y89 = y89, z89 = z89, w89 = w89,  
    x90 = x90, y90 = y90, z90 = z90, w90 = w90,  
    x91 = x91, y91 = y91, z91 = z91, w91 = w91,  
    x92 = x92, y92 = y92, z92 = z92, w92 = w92,  
    x93 = x93, y93 = y93, z93 = z93, w93 = w93,  
    x94 = x94, y94 = y94, z94 = z94, w94 = w94,  
    x95 = x95, y95 = y95, z95 = z95, w95 = w95,  
    x96 = x96, y96 = y96, z96 = z96, w96 = w96,  
    x97 = x97, y97 = y97, z97 = z97, w97 = w97,  
    x98 = x98, y98 = y98, z98 = z98, w98 = w98,  
    x99 = x99, y99 = y99, z99 = z99, w99 = w99,  
    x100 = x100, y100 = y100, z100 = z100, w100 = w100,  
    x101 = x101, y101 = y101, z101 = z101, w101 = w101,  
    x102 = x102, y102 = y102, z102 = z102, w102 = w102,  
    x103 = x103, y103 = y103, z103 = z103, w103 = w103,  
    x104 = x104, y104 = y104, z104 = z104, w104 = w104,  
    x105 = x105, y105 = y105, z105 = z105, w105 = w105,  
    x106 = x106, y106 = y106, z106 = z106, w106 = w106,  
    x107 = x107, y107 = y107, z107 = z107, w107 = w107,  
    x108 = x108, y108 = y108, z108 = z108, w108 = w108,  
    x109 = x109, y109 = y109, z109 = z109, w109 = w109,  
    x110 = x110, y110 = y110, z110 = z110, w110 = w110,  
    x111 = x111, y111 = y111, z111 = z111, w111 = w111,  
    x112 = x112, y112 = y112, z112 = z112, w112 = w112,  
    x113 = x113, y113 = y113, z113 = z113, w113 = w113,  
    x114 = x114, y114 = y114, z114 = z114, w114 = w114,  
    x115 = x115, y115 = y115, z115 = z115, w115 = w115,  
    x116 = x1
```

```
flat_samples, labels=labels, truths=[m_true, b_true, np.log(f_true)], levels=(0.90,), quantiles=(0.06, 0.
);
```



```
plt.errorbar(x, y, yerr=yerr, fmt='k', capsize=0)
plt.plot(x, m_true * x + b_true, "k", label="truth")
plt.legend(fontsize=14)
plt.xlabel("z")
plt.ylabel("rgas");
```

The plot displays the gas radius (r_{gas}) on the y-axis against redshift (z) on the x-axis. The y-axis ranges from 0.200 to 0.225. The x-axis ranges from 0 to 1.0. A legend in the top right corner indicates the 'truth' line. The plot shows several data points with vertical error bars, representing the uncertainty in the gas radius measurements. The data points are plotted as black circles with error bars, and the 'truth' line is a solid black line.



```
In [92]: from IPython.display import display, Math

for i in range(ndim):
    mcmc = np.percentile(flat_samples[:, i], [6, 50, 90])
    q = np.diff(mcmc)
    txt = "\mathrm{{{3}}} = \{0:.3f\}_{- \{1:.3f\}}^{\{2:.3f\}}"
    txt = txt.format(mcmc[1], q[0], q[1], labels[i])
    txt1 = "\mathrm{{{3}}} = \{0:.3f\}_{- \{1:.3f\}}^{\{2:.3f\}}"
    display(Math(txt))
```

$$\begin{aligned} m &= -0.008^{+0.014}_{-0.016} \\ b &= 0.121^{+0.009}_{-0.011} \\ \log(f) &= -1.977^{+0.173}_{-0.242} \end{aligned}$$

- ```
print("T0 = {0:.3f}".format(-0.008))
```