K.E. Whitaker Wednesday, September 7, 2016

Tutorial for reading and manipulating catalogs in Python[1]

First, download the data

The catalogs that you will need for this exercise can be downloaded here:
https://www.dropbox.com/sh/le0d8971tmufcqx/AAAxiVdo2P_pg63OrHkdudr7a?dl=0

I've placed them in a working directory "merged_catalogs" at the same level as a sister directory where you'll work on this tutorial (as listed below), so adjust the second input definition for `catalog_path` accordingly depending on where you save the catalogs.

- You can read catalogs in one of two ways:
  - Read the .fits format files
  - Read the original ascii files, either one by one or use a master file. This is a slower option, but may be optimal if you have memory limitations.

Import a few utilities from astropy

If you don't have astropy, you can get it one of several ways. I recommend installing Ureka (http://ssb.stsci.edu/ureka/), but you can also get it from Anaconda (https://store.continuum.io/cshop/anaconda/), or by installing using `pip install astropy` or by cloning the astropy github repository (https://github.com/astropy/astropy) and installing from source.

Once it's installed, you can type `ipython notebook` to open your notebook and first import a few handy utilities

```
In [1]: from astropy.table import Table, Column, join
        from astropy.coordinates import SkyCoord
        from astropy.table import Column
        from numpy import *
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Create a variable that points to the directory holding the catalogs. The "../" tells you to look up one directory. You may prefer to write the absolute path name instead. NOTE: you are explicitly defining the location of the file for which you want to read in to python. It therefore matters what you type. If you did not "mkdir merge_catalogs" and move your downloaded files here, this step will give you an error. Think carefully!

```
In [2]: catalog_path="../merged_catalogs/" # Trailing / is required
```

Read in the 3D-HST COSMOS catalog

First we'll read in the 3D-HST COSMOS catalog from the FITS formatted file.

```
In [3]: cosmos = Table.read(catalog_path+'cosmos.zbest.v4.1.5.fits')
```

You can inspect a few different ways. Type `help(cosmos)` for a list of methods. Below are a few examples…

```
In [4]: print cosmos

   ID [33879]                ID_SPEC [33879]               ... GALFIT_RATIO_F160W [33879]
-------------- -------------------------------------------- ... --------------------------
 1.0 .. 33879.0 00000                .. 00000               ...          -99.0 .. 0.91667
```

[1] This python tutorial has been adapted from the original CANDELS tutorial by H. Ferguson

```
In [5]: cosmos.colnames  # List all of the column names

Out [5]:
['ID',
 'ID_SPEC',
 'FIELD',
 'RA',
 'DEC',
 'X',
 'Y',
 'Z_BEST',
 'Z_TYPE',
 'Z_SPEC',
 'Z_PHOT',
 'Z_GRISM',
 'F_F140',
 'E_F140',
 'MAG_F140',
 'F_F160',
 'E_F160',
 'MAG_F160',
 'LMASS',
 'AV',
 'LAGE',
 'LSFR',
 'LSSFR',
 'FLUX_RADIUS',
 'UV',
 'VJ',
 'UVJ',
 'RE_F140',
 'E_RE_F140',
 'N_F140',
 'E_N_F140',
 'Q_F140',
 'E_Q_F140',
 'RE_F160',
 'E_RE_F160',
 'N_F160',
 'E_N_F160',
 'Q_F160',
 'E_Q_F160',
 'RE_F125',
 'E_RE_F125',
 'N_F125',
 'E_N_F125',
 'Q_F125',
 'E_Q_F125',
 'MAG_F814',
 'USE',
 'FLAG_F125',
 'FLAG_F140',
 'FLAG_F160',
 'GALFIT_RATIO_F125W',
 'GALFIT_RATIO_F140W',
 'GALFIT_RATIO_F160W']
```

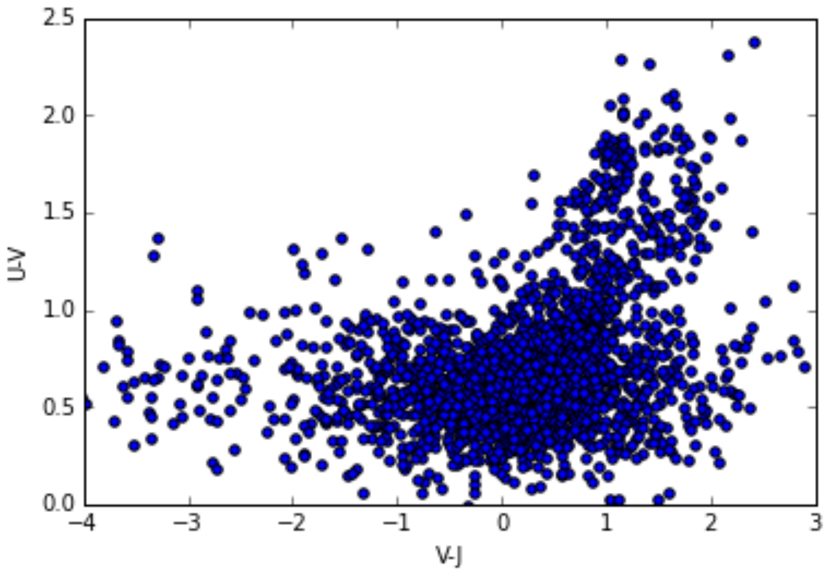
Select galaxies and plot columns

Let's work in AB magnitudes for convenience (defining a function to do the conversion from the catalog fluxes, which are AB magnitudes with a zero point of 25). We'll select galaxies with good photometry (`use==0`), with `z_best` redshifts 1.8<z<2.2, and with H_AB < 26.5 (`MAG_F160<26.5`), and we'll plot a rest-frame UVJ diagram.

```
# Note:  (1) the parentheses () are required here due to the operator precedence
in Python numpy, (2) the keywords are case sensitive, and (3) make sure if copy
and pasting that the quotations are simple (you may need to delete and retype

In [6]: selection= (cosmos['USE'] == 1) & (cosmos['Z_BEST']>1.8) &
        (cosmos['Z_BEST']<2.2) & (cosmos['MAG_F160']<26.5)
```

```
In [7]: uv = cosmos['UV'][selection]
        vj = cosmos['VJ'][selection]
        print len(uv)  # How many data points do we have?
2498
```

```
# Plot the UVJ diagram

In [8]: plt.scatter(vj,uv)
        plt.xlim(-4,3)
        plt.ylim(0,2.5)
        plt.xlabel('V-J')
        plt.ylabel('U-V')
```



For fun, let's color-code the points with specific star-formation rate estimated from the unobscured (ultraviolet) plus obscured (infrared) star formation rates presented in Whitaker et al. (2014). These star formation rates are in a separate catalog, so first we need to read in that (ascii) catalog. Next, we will make a histogram to determine the range of sSFR, then finally we will make the UVJ diagram.

```
In [9]: from astropy.io import ascii
```

```
    # There are number of lines of header information in the SFR file; note that we
    define where the header (and column names) are located, as well as where to
    start reading the data.  Delimiter signifies how columns are separated.

In [10]: cosmos_sfr = ascii.read(catalog_path+'cosmos_3dhst.v4.1.5.zbest.sfr',
         data_start=0,header_start=0,delimiter=' ')
```

```
    # Note that the SFR catalog has the same number of entries as the earlier
    catalog, with 33879 rows (see earlier print statement).  If these values are not
    identical, there was a problem reading in the catalog!

In [11]: print cosmos_sfr
Out [11]:

     id    sfr    sfr_IR    sfr_UV    ... ef24tot    L_1600          L_2800       beta
    ----- ------- ------- ----------- ... ------- -------------- -------------- -----
        1  214.34  206.45      7.8873 ...   29.22   3306500000.0   21927000000.0  2.37
        2  191.43  181.25      10.182 ...   33.61   8262000000.0   28308000000.0  1.19
        3  2.4763  1.3982      1.0782 ...   31.62   1622200000.0    2997400000.0 -0.06
        4  306.65  304.82      1.8354 ...   41.84   7822800000.0    5102500000.0 -1.78
        5   -99.0   -99.0   0.0036845 ...   27.12        78668.0      10243000.0  8.89
        6   -99.0   -99.0   0.0076811 ...    23.8     14203000.0      21354000.0  1.11
        7   -99.0   -99.0     0.20115 ...   33.91    139400000.0     559220000.0  1.06
        8   -99.0   -99.0    0.042787 ...   33.25     90102000.0     118950000.0 -0.37
        9   -99.0   -99.0    0.085605 ...    25.6    117150000.0     237990000.0 -0.01
       10   -99.0   -99.0      0.3596 ...   36.49    219220000.0     999730000.0  2.03
      ...     ...     ...         ... ...     ...            ...             ...   ...
    33869   -99.0   -99.0     0.71492 ...   15.93   3428500000.0    1987500000.0 -1.87
    33870  460.38  251.64      208.74 ...   19.82 247870000000.0 580310000000.0  0.58
    33871 0.80725 0.61849     0.18877 ...   23.09    277780000.0     524790000.0  0.23
    33872   -99.0   -99.0     0.36034 ...   27.08   1396900000.0    1001800000.0 -1.53
    33873  1206.7  1146.5      60.209 ...   19.13 152750000000.0 167390000000.0 -0.66
    33874   -99.0   -99.0      35.013 ...   13.06  53427000000.0   97340000000.0  0.19
    33875  2.5188  2.5182  0.00056864 ...   15.55        0.83809       1580900.0 26.66
    33876   3.594  3.5706    0.023439 ...   20.39        42053.0      65161000.0 14.62
    33877   -99.0   -99.0  1.0616e-11 ...   21.09         0.0277        0.029512 -99.0
    33878   -99.0   -99.0       -99.0 ...   22.63          -99.0           -99.0 -99.0
    33879   -99.0   -99.0      48.627 ...   23.89  82957000000.0 135190000000.0   0.0
    Length = 33879 rows
```

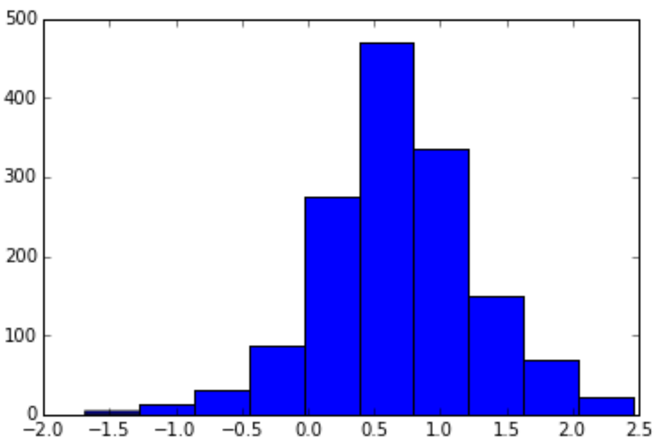
```
In [12]: cosmos_sfr.colnames
Out [12]:
         ['id',
          'sfr',
          'sfr_IR',
          'sfr_UV',
          'L_IR',
          'L_UV',
          'flag',
          'z_best',
          'z_type',
          'f24tot',
          'ef24tot',
          'L_1600',
          'L_2800',
          'beta']
```

```
  # Now lets update the selection to only include galaxies with a SFR > 0, as we
  will be taking the logarithm of SFR.

In [13]:cosmos_selection = (cosmos['USE'] == 1) & (cosmos['Z_BEST']>1.8) &
        (cosmos['Z_BEST']<2.2) & (cosmos['MAG_F160']<26.5)  &
        (cosmos_sfr['sfr']>0)
```

```
  # How many galaxies did we select?  Note that the ascii structure “cosmos_sfr”
  has a different shape to the fits structure “cosmos”.  For this reason, we
  include [0] to select the full column.  If they had the same shape, ideally we
  would want to use “join” to create a single structure.

In [14]:print cosmos_sfr[‘sfr’][cosmos_selection[0]]
Out [14]:
         sfr
        ------
        147.31
        11.593
        17.097
        14.408
        42.476
        15.527
        7.6612
        10.627
        54.593
        41.978
           ...
        26.113
        11.914
        2.9159
         12.86
        25.767
        8.7395
        9.8323
         8.593
        10.616
        96.386
        5.9362
        Length = 1455 rows
```

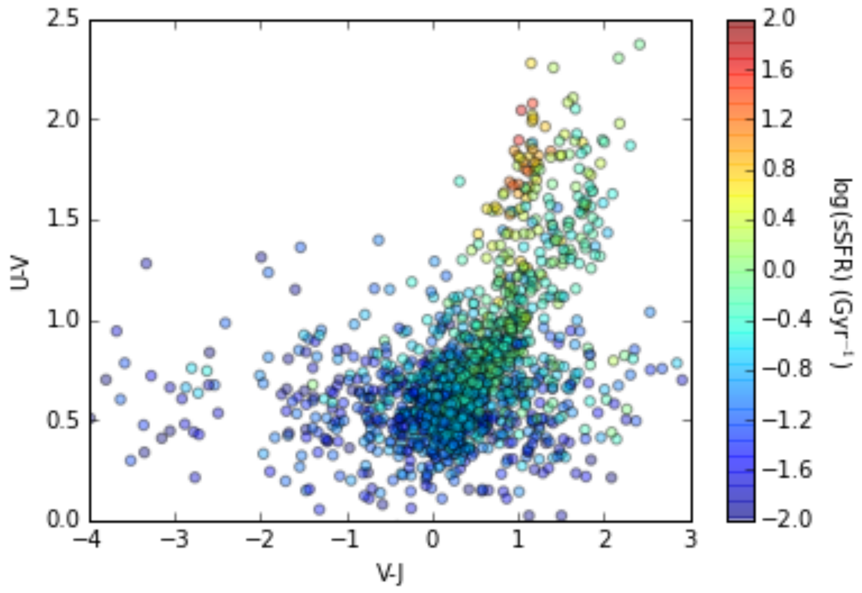
Now that we have the SFR data from the separate ascii file, we want to color-code each data point in our original UVJ diagram by this SFR per unit stellar mass, or the specific SFR (sSFR). First we need to make a histogram to learn the range in sSFR, and next remake the UVJ diagram.

```
  # sSFR = SFR/M*, so log(sSFR) = log(SFR)-log(M*).  As we want the sSFR in
  units of Gyr^-1, we add 9 in the logarithmic units.

In [15]: ssfr_cosmos = log10(cosmos_sfr[‘sfr’][cosmos_selection[0]])-
         cosmos[‘LMASS’][cosmos_selection]+9.

         n,bins,patches = plt.hist(ssfr)
```



# Plot the UVJ diagram. The color map 'jet' runs from red to blue, so we need to invert the sSFR to make the quenched galaxies red. We also need to redefine the U-V and V-J colors, as we reduced the selection by requiring a measured SFR and we want to avoid confusion. Alpha adds transparency to the points, after which we add a colorbar and update the x and y limits and labels. The label pad shifts the colorbar label right so it isn't overlapping with the numbers.

```
In [16]: uv_cosmos = cosmos['UV'][cosmos_selection]
         vj_cosmos = cosmos['VJ'][cosmos_selection]
         plt.scatter(vj_cosmos,uv_cosmos,c=-ssfr, vmin=-2,vmax=2,cmap='jet',alpha=0.4)
         cbar = plt.colorbar()  # Adds a colorbar
         cbar.set_label('log(sSFR) (Gyr$^{-1}$)',rotation=270,labelpad=20)
         plt.xlim(-4,3)
         plt.ylim(0,2.5)
         plt.xlabel('V-J')
         plt.ylabel('U-V')
```



Read in and combine catalogs from multiple fields

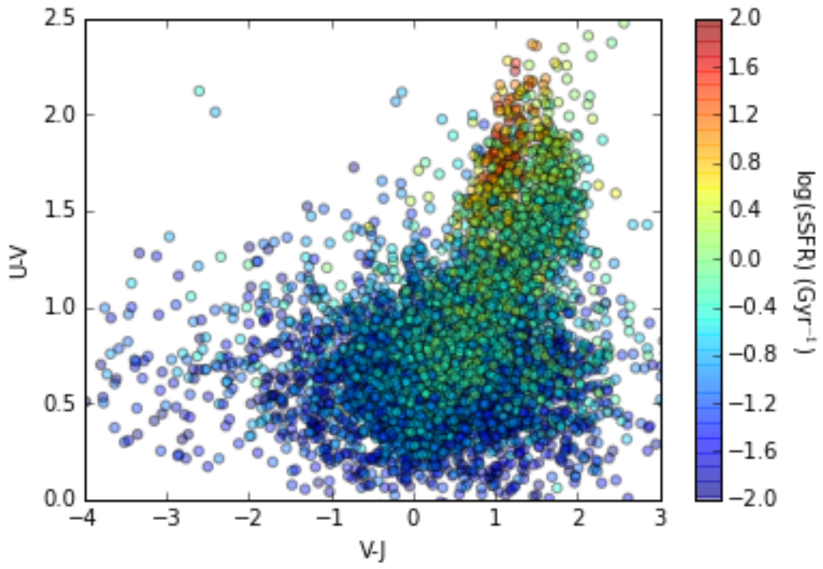
So far we have only read in data from the COSMOS extragalactic field, but the 3D-HST treasury program includes catalogs from 4 other extragalactic fields: AEGIS, GOODS-S, GOODS-N and UDS. Next we will read in the catalogs from the other fields and learn how to combine the data structures into one. We can then repeat our selection and make the UVJ diagram color-coded by sSFR for the entire sample.

```
    # We will start by combining COSMOS with AEGIS.

In [17]:aegis = Table.read(catalog_path+’aegis.zbest.v4.1.5.fits')
        aegis_sfr = ascii.read(catalog_path+’aegis_3dhst.v4.1.5.zbest.sfr’,
        data_start=0,header_start=0,delimiter=' ‘)
        aegis_selection= (aegis['USE'] == 1) & (aegis['Z_BEST']>1.8) &
        (aegis['Z_BEST']<2.2) & (aegis['MAG_F160']<26.5)  & (aegis_sfr['sfr']>0)
        uv_aegis = aegis['UV'][aegis_selection]
        vj_aegis = aegis['VJ'][aegis_selection]
        ssfr_aegis = log10(aegis_sfr['sfr'][aegis_selection[0]])-aegis['LMASS']
        [aegis_selection]+9.

        uv = concatenate([uv_aegis.data,uv_cosmos.data])
        vj = concatenate([vj_aegis.data,vj_cosmos.data])
        ssfr = concatenate([ssfr_aegis.data,ssfr_cosmos.data])
```

```
    # Now that we have combined two, we can combine all five fields.

In [18]:uds = Table.read(catalog_path+'uds.zbest.v4.1.5.fits')
        uds_sfr = ascii.read(catalog_path+’uds_3dhst.v4.1.5.zbest.sfr',
        data_start=0,header_start=0,delimiter=' ')
        uds_selection= (uds['USE'] == 1) & (uds['Z_BEST']>1.8) & (uds['Z_BEST']<2.2)
        & (uds['MAG_F160']<26.5)  & (uds_sfr['sfr']>0)
        uv_uds = uds['UV'][uds_selection]
        vj_uds = uds['VJ'][uds_selection]
        ssfr_uds = log10(uds_sfr['sfr'][uds_selection[0]])-uds['LMASS']
        [uds_selection]+9.

        goodss = Table.read(catalog_path+'goodss.zbest.v4.1.5.fits')
        goodss_sfr = ascii.read(catalog_path+’goodss_3dhst.v4.1.5.zbest.sfr’,
        data_start=0,header_start=0,delimiter=' ')
        goodss_selection= (goodss['USE'] == 1) & (goodss['Z_BEST']>1.8) &
        (goodss['Z_BEST']<2.2) & (goodss['MAG_F160']<26.5)  & (goodss_sfr['sfr']>0)
        uv_goodss = goodss['UV'][goodss_selection]
        vj_goodss = goodss['VJ'][goodss_selection]
        ssfr_goodss = log10(goodss_sfr['sfr'][goodss_selection[0]])-goodss['LMASS']
        [goodss_selection]+9.

        goodsn = Table.read(catalog_path+'goodsn.zbest.v4.1.5.fits')
        goodsn_sfr = ascii.read(catalog_path+’goodsn_3dhst.v4.1.5.zbest.sfr',
        data_start=0,header_start=0,delimiter=' ')
        goodsn_selection= (goodsn['USE'] == 1) & (goodsn['Z_BEST']>1.8) &
        (goodsn['Z_BEST']<2.2) & (goodsn['MAG_F160']<26.5)  & (goodsn_sfr['sfr']>0)
        uv_goodsn = goodsn['UV'][goodsn_selection]
        vj_goodsn = goodsn['VJ'][goodsn_selection]
        ssfr_goodsn = log10(goodsn_sfr['sfr'][goodsn_selection[0]])-goodsn['LMASS']
        [goodsn_selection]+9.

        uv = concatenate([uv_cosmos.data,uv_aegis.data,
        uv_uds.data,uv_goodss.data,uv_goodsn.data])
        vj = concatenate([vj_cosmos.data,vj_aegis.data,
        vj_uds.data,vj_goodss.data,vj_goodsn.data])
        ssfr = concatenate([ssfr_cosmos.data,ssfr_aegis.data,
        ssfr_uds.data,ssfr_goodss.data, ssfr_goodsn.data])
```

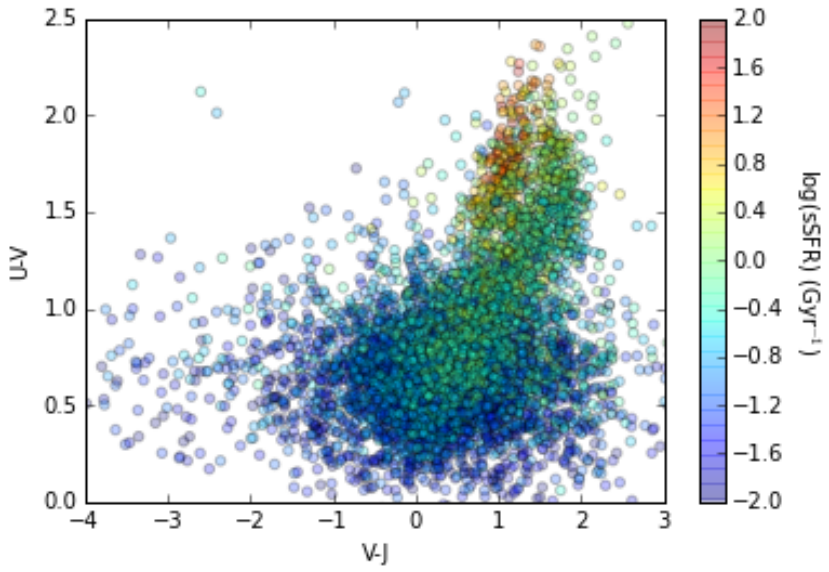
Take note that if you need to re-run these commands either to de-bug or for other reasons, this definition of the U-V, V-J and sSFR are cumulative. Therefore, you may want to edit the text to return to the individual definitions for each field (e.g., uv_field.data, where field = cosmos, aegis, uds, goodss, or goodsn). If you are concerned this has occurred, one way to check is to type `print len(vj), len(uv), len(ssfr)` and ensure that the lengths are each equal to 8227 galaxies.

```
  # The final UVJ diagram for all 5 3D-HST fields

In [19]:plt.scatter(vj,uv,c=-ssfr,vmin=-2,vmax=2,cmap='jet',alpha=0.4)
        cbar = plt.colorbar()  # Adds a colorbar
        cbar.set_label('log(sSFR) (Gyr$^{-1}$)',rotation=270,labelpad=20)
        plt.xlim(-4,3)
        plt.ylim(0,2.5)
        plt.xlabel('V-J')
        plt.ylabel('U-V')
```



Notice that we have reversed the color-bar within the figure but not in the actual plotted values (as red should signify the quiescent galaxies with the lowest sSFRs). Instead of changing the sSFR data itself, which will make the colorbar axis labels reversed, we can instead use the reverse colormap by appending '_r' at the end of the colormap name.

```
  # The final UVJ diagram for all 5 3D-HST fields (with correct colorbar,
  transparency slightly decreased)

In [20]:plt.scatter(vj,uv,c=ssfr,vmin=-2,vmax=2,cmap='jet_r',alpha=0.25)
        cbar = plt.colorbar()  # Adds a colorbar
        cbar.set_label('log(sSFR) (Gyr$^{-1}$)',rotation=270,labelpad=20)
        plt.xlim(-4,3)
        plt.ylim(0,2.5)
        plt.xlabel('V-J')
        plt.ylabel('U-V')
```



What do we learn from this plot?

Both the rest-frame U-V and V-J colors and the sSFRs encapsulate important information about the stellar populations of galaxies across all of cosmic history. We see in this figure that those galaxies with the reddest U-V colors (higher value = redder color) have the lowest sSFRs; we can cleanly select based on their U-V and V-J colors the quiescent population with passively evolution stellar populations. Conversely, those galaxies with the bluest U-V colors have the highest sSFRs.

The inverse of the sSFR gives the rough timescale it would take for the galaxy to double in mass. We see that the bluest galaxies have 1/sSFR ~ 1/10^2 Gyr, or ~10 Myr to double their stellar mass. On the other hand, quiescent galaxies with 1/sSFR ~1/10^{-2} Gyr is equivalent to 100 Gyr. These galaxies are forming so few new stars relative to their total stellar mass, that it would take until 7 times the current age of the Universe for their stellar mass to double (assuming the SFR remains constant over this entire time).

Check out the rest-frame colors presented as a function of redshift in Figure 17 of Whitaker et al. 2011: http://iopscience.iop.org/article/10.1088/0004-637X/735/2/86/pdf. There is clear bimodality in the rest-frame colors that allows us to separate quiescent galaxies from the actively star-forming galaxies across >85% of cosmic history. A cosmological redshift of z=3.5 corresponds to a lookback time of 11.9 billion years, when the Universe was only 1.8 years old. The implications of this figure are important: when the Universe itself was less than 2 billion years old, the most massive galaxies had already formed the bulk of their stars and quenched star formation. These observations challenge theoretical models, as the timescales are incredibly short to form such a vast amount of stars. Astronomers are still working to understand the build-up of this quiescent population in the early Universe.