# CompBioBase - Live-Examples

## Live-examples (LE) development guidelines

Folder TEMPLATE contains a template, use it to start the development of a new live-example, or reformat an existing example.

### Naming and version control

Every LE release which has substantial changes should be assigned a version number which sould appera in the header of README.md. A history of changes should be provided in HISTORY.md file together with hashes of commits that correspond to the current and previous versions. Folder name should be concise but suggestive of the example nature and start whith a letter code from this list depending on the LE topic. If LE is forked to adjust it to slightly different conditions (eg. software or hardware version) the new folder name should be derived by appending a corresonding descriptor (eg. MD_nucleosome_NAMD_v1.9_wCUDA). For such forked LE the information about their parent (sister) LE should be provided in the header of README.md and the corresponding versions have to be specified.

### Tags

LE are organized with the help of tags (keywords) from the dictionary listed in tags_list.md. Every LE folder should contain tags.md file which list tags from the dictionary one per line.

### LE status

If LE adheres to the strict guidelines described below its status is READY, otherwise it is RAW. The status should be stated in the header of README.md.

### Live-example components and directory structure

**README.md** The README.md file is the main file describing the live-example, theoretical background, goals, prerequisitives (including software), contents of the live-example, step-by-step instructions and anticipated outcomes. Normally, it should have following sections:

The files should be written using strick Markdown syntax (not GFM?) in order to be understandable by pandoc.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import argparse
import csv
from pylab import *
from prody import *
#import pickle
EVNUM=40

def main():



    matplotlib.rc('font',**{'family':'sans-serif','sans-serif':['Helvetica'],'size':14})
    plt.rcParams['ps.useafm'] = True
    # rc('font',**{'family':'sans-serif','sans-serif':['Helvetica']})
    # plt.rcParams['pdf.fonttype'] = 82
    # matplotlib.rc('font', **font)
    alpha_core='((segment CHA CHE and (resid 64 to 78 or resid 86 to 114 or resid 121 to 131
    #                                          15                 29                    11



        # sp.plot(np.arange(0, float(edaHF.numAtoms()),1), calcSqFlucts(edaHF[i]), label="%
        # sp.axvspan(HFind['H3A'][0],HFind['H3A'][-1] , color='blue', alpha=0.2)
        # sp.axvline(x=HFind['H3A'][1],color='k')
        # sp.axvline(x=HFind['H3A'][2],color='k')
        # sp.axvspan(HFind['H4B'][0],HFind['H4B'][-1] , color='green', alpha=0.2)
        # sp.axvline(x=HFind['H4B'][1],color='k')
        # sp.axvline(x=HFind['H4B'][2],color='k')
        # sp.axvspan(HFind['H2AC'][0],HFind['H2AC'][-1] , color='yellow', alpha=0.2)
        # sp.axvline(x=HFind['H2AC'][1],color='k')
        # sp.axvline(x=HFind['H2AC'][2],color='k')
        # sp.axvspan(HFind['H2BD'][0],HFind['H2BD'][-1] , color='red', alpha=0.2)
        # sp.axvline(x=HFind['H2BD'][1],color='k')
        # sp.axvline(x=HFind['H2BD'][2],color='k')

        # sp.axvspan(HFind['H3E'][0],HFind['H3E'][-1] , color='blue', alpha=0.2)
        # sp.axvline(x=HFind['H3E'][1],color='k')
        # sp.axvline(x=HFind['H3E'][2],color='k')
        # sp.axvspan(HFind['H4F'][0],HFind['H4F'][-1] , color='green', alpha=0.2)
        # sp.axvline(x=HFind['H4F'][1],color='k')
        # sp.axvline(x=HFind['H4F'][2],color='k')
        # sp.axvspan(HFind['H2AG'][0],HFind['H2AG'][-1] , color='yellow', alpha=0.2)
        # sp.axvline(x=HFind['H2AG'][1],color='k')
        # sp.axvline(x=HFind['H2AG'][2],color='k')
        # sp.axvspan(HFind['H2BH'][0],HFind['H2BH'][-1] , color='red', alpha=0.2)
        # sp.axvline(x=HFind['H2BH'][1],color='k')
        # sp.axvline(x=HFind['H2BH'][2],color='k')


#Now let's do covariance
    #let's dump it
    #pickle.dump(calcCrossCorr(edaHF), open( "../analysis_data/hfolds_var_covar.p", "wb" ) 
    cross_covar = pickle.load( open( "../analysis_data/dnabb_covar.p", "rb" ) )
    # arange = np.arange(1656)
    # cross_correlations = np.zeros((arange[-1]+2, arange[-1]+2))
    # cross_correlations[arange[0]+1:,arange[0]+1:] = cross_corr
    cross_covar=cross_covar[0:(cross_covar.shape[0]/2),:]
    print cross_covar.shape
    fig4=plt.figure(figsize={15,7})
    sp=fig4.add_subplot(111)
    cax=sp.imshow(cross_covar,interpolation='none')
    cp=fig4.colorbar(cax,shrink=0.75)
    cp.set_label("$A^2$")
    #sp.set_axis([arange[0]+0.5, arange[-1]+1.5, arange[0]+0.5, arange[-1]+1.5])
    sp.set_title('Variance-covariance')
    sp.set_xlabel('Super helix location (SHL)')
    sp.set_ylabel('Super helix location (SHL)')
    sp.set_title("Variance-covariance matrix for DNA backbone.")

    # sp.set_xlim(0,100)

    SHL_loc=np.array([2,12,22,32,42,52,62,72,82,92,102,112,122,132,142])
    SHL_loc_ticks=np.array([-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7])

    SHL_loc_ticksx=np.concatenate((SHL_loc_ticks,SHL_loc_ticks))
    SHL_locx=np.concatenate((SHL_loc,SHL_loc+146))

    sp.set_xticks(SHL_locx)
    sp.set_xticklabels(SHL_loc_ticksx)

    sp.set_yticks(SHL_loc)
    sp.set_yticklabels(SHL_loc_ticks)
    sp.tick_params( labelright=True)

    sp.axvline(x=145.5,color='k')
    # sp.axvline(x=146,color='k')

    sp.annotate('Chain I', xy=(.2, .06), xycoords='figure fraction', horizontalalignment='le
    sp.annotate('Chain J', xy=(.59, .06), xycoords='figure fraction', horizontalalignment='l
    sp.annotate('Chain I', xy=(.817, .55), xycoords='figure fraction', horizontalalignment='

    fig4.tight_layout()
    fig4.savefig("../analysis_data/pca_dnabb_covar.png",dpi=(200))

    plt.show()
    # savefig('../analysis_data/pca_plot.png',dpi=(200))


    # writeNMD('../analysis_data/eda_DNA.nmd', edaDNA[:5], pdb.select("name P"))




if __name__ == '__main__':
    main()
```