In [1]:
```
# activiate inline plotting
%matplotlib inline
```

In [1]:
```
import sys
import scipy as SP
import pylab as PL
from matplotlib import cm
import h5py
#import scLVM
sys.path.append('./../scLVM')
from scLVM import scLVM
```

Import data and filter sets of genes

In [2]:
```
data = './../data/Tcell/normCountsMMus_final.h5f'
f = h5py.File(data,'r')
Y = f['LogNcountsMmus'][:]                    # gene expression matrix
tech_noise = f['LogVar_techMmus'][:]          # technical noise
genes_het_bool=f['genes_heterogen'][:]        # index of heterogeneous gen
geneID = f['gene_names'][:]                # gene names
cellcyclegenes_filter = SP.unique(f['cellcyclegenes_filter'][:].ravel()
cellcyclegenes_filterCB600 = f['ccCBall_gene_indices'][:].ravel() -1
```

In [3]:
```
# filter cell cycle genes
idx_cell_cycle = SP.union1d(cellcyclegenes_filter,cellcyclegenes_filter
Ymean2 = Y.mean(0)**2>0
idx_cell_cycle_noise_filtered = SP.intersect1d(idx_cell_cycle,SP.array(
Ycc = Y[:,idx_cell_cycle_noise_filtered]
```

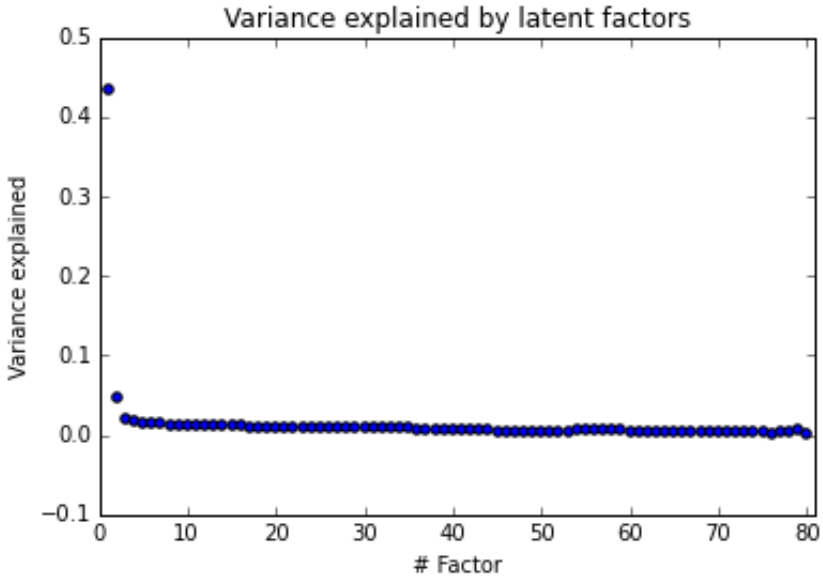
Fit GPLVM to the data

In [27]:
```
k = 80                      # number of latent factors
out_dir = './cache'         # folder where results are cached
file_name = 'Kcc.hdf5'      # name of the cache file
recalc = True               # recalculate X and Kconf
use_ard = True              # use automatic relevance detection
sclvm = scLVM(Y)
X_ARD,Kcc_ARD,varGPLVM_ARD = sclvm.fitGPLVM(idx=idx_cell_cycle_noise_fi
```

In [28]:
```
#Plot variance contributions from ARD
plt = PL.subplot(1,1,1)
PL.title('Variance explained by latent factors')
PL.scatter(SP.arange(k)+1,varGPLVM_ARD['X_ARD'])
PL.xlim([0,k+1])
```

```
PL.xlabel('# Factor')
PL.ylabel('Variance explained')
```

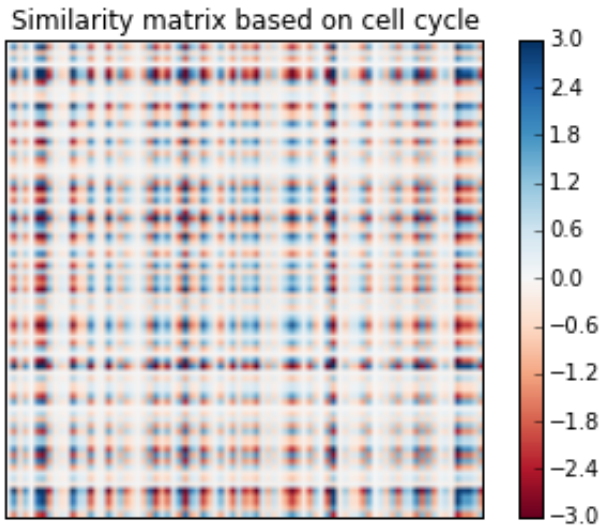
Out[28]: <matplotlib.text.Text at 0x10fbd3850>



In [29]:
```
#Fit rank 1 covariance matrix (large gap between first and second laten
X,Kcc,varGPLVM = sclvm.fitGPLVM(idx=idx_cell_cycle_noise_filtered,k=1,o
```

In [30]:
```
#Plot inferred similarity matrix
plt = PL.subplot(1,1,1)
PL.title('Similarity matrix based on cell cycle')
PL.imshow(Kcc,cmap=cm.RdBu,vmin=-3,vmax=+3)
PL.colorbar()
plt.set_xticks([])
plt.set_yticks([])
```

Out[30]: []



Variance Decomposition and Corrections

In [25]:
```
# considers only heterogeneous genes
Ihet = genes_het_bool==1
Y    = Y[:,Ihet]
tech_noise = tech_noise[Ihet]
geneID = geneID[Ihet]
```

In [8]:
```
i0 = 0    # gene from which the analysis starts
i1 = 50   # gene at which the analysis ends

# define sclvm
sclvm = scLVM(Y,geneID=geneID,tech_noise=tech_noise)

# fit the model from i0 to i1
sclvm.varianceDecomposition(K=Kcc,i0=i0,i1=i1)
```

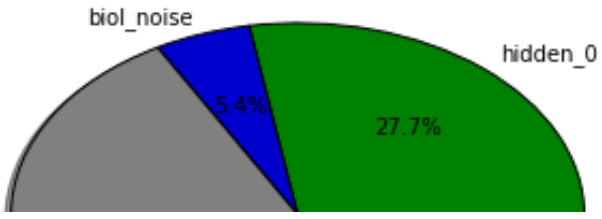
In [9]:
```
normalize=True    # variance components are normalizaed to sum up to on

# get variance components
var, var_info = sclvm.getVarianceComponents(normalize=normalize)
var_filtered = var[var_info['conv']] # filter out genes for which vd ha

# get corrected expression levels
Ycorr = sclvm.getCorrectedExpression()
```

In [10]:
```
#normalize variance component and average across genes
var_mean = var_filtered.mean(0)
colors = ['Green','MediumBlue','Gray']
PL.pie(var_mean,labels=var_info['col_header'],autopct='%1.1f%%',colors=
       shadow=True, startangle=0)
```

Out[10]:
```
([<matplotlib.patches.Wedge at 0x111c2a950>,
  <matplotlib.patches.Wedge at 0x111c38cd0>,
  <matplotlib.patches.Wedge at 0x111c45fd0>],
 [<matplotlib.text.Text at 0x111c38390>,
  <matplotlib.text.Text at 0x111c456d0>,
  <matplotlib.text.Text at 0x111c529d0>],
 [<matplotlib.text.Text at 0x111c38890>,
  <matplotlib.text.Text at 0x111c45b90>,
  <matplotlib.text.Text at 0x111c52e90>])
```



Fitting Linear Mixed Model for correlations

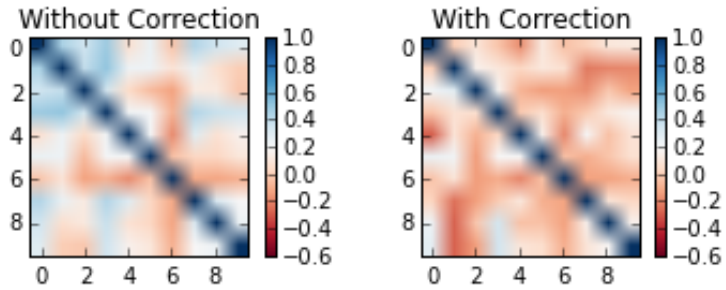
In [11]:
```
i0 = 0      # gene from which the analysis starts
i1 = 10     # gene to which the analysis ends

# fit lmm without correction
pv0,beta0,info0 = sclvm.fitLMM(K=None,i0=i0,i1=i1,verbose=True)
# fit lmm with correction
pv1,beta1,info1 = sclvm.fitLMM(K=Kcc,i0=i0,i1=i1,verbose=True)
```

```
.. fitting gene 0
.. fitting gene 1
.. fitting gene 2
.. fitting gene 3
.. fitting gene 4
.. fitting gene 5
.. fitting gene 6
.. fitting gene 7
.. fitting gene 8
.. fitting gene 9
.. fitting gene 0
.. fitting gene 1
.. fitting gene 2
.. fitting gene 3
.. fitting gene 4
.. fitting gene 5
.. fitting gene 6
.. fitting gene 7
.. fitting gene 8
.. fitting gene 9
```

In [28]:
```
PL.subplot(2,2,1)
PL.title('Without Correction')
PL.imshow(beta0[:,i0:i1],cmap=cm.RdBu,vmin=-0.6,vmax=+1)
PL.colorbar()
plt.set_xticks([])
plt.set_yticks([])
PL.subplot(2,2,2)
PL.title('With Correction')
PL.imshow(beta1[:,i0:i1],cmap=cm.RdBu,vmin=-0.6,vmax=+1)
PL.colorbar()
plt.set_xticks([])
plt.set_yticks([])
```

Out[28]: 0


Without Correction
With Correction
0
2
4
6
8
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6