In [17]:

```
# define the functions

%pylab nbagg
from tvb.simulator.lab import *
from tvb.datatypes.time_series import TimeSeriesRegion
import numpy as np
import time as tm
import matplotlib.pyplot as plt
import sys
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [ ]:

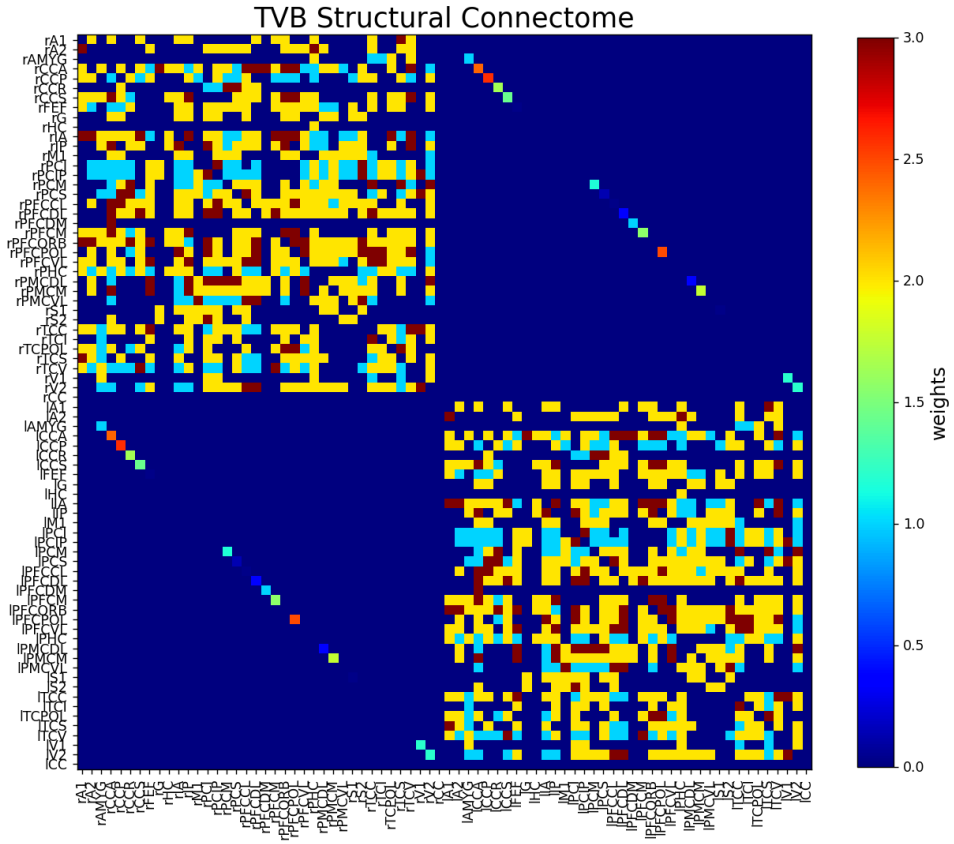
```
# Connectivity
We start by loading and visualizing the structural connectivity matrix that represen
we can then alter the speed of signal propagation through the network.
```

In [8]:

```
# Initialise the Connectivity.

con = connectivity.Connectivity.from_file('connectivity_76.zip')
nregions = len(con.region_labels)                                    #number of regions
con.weights = con.weights - con.weights * eye((nregions))            #remove self-connect
con.speed = np.array([sys.float_info.max])                           #set conduction spee
con.configure()

# Visualization.
figure(figsize=(12,12))
imshow(con.weights, interpolation='nearest', aspect='equal', cmap='jet')
title('TVB Structural Connectome', fontsize=20)
xticks(range(0, nregions), con.region_labels, fontsize=10, rotation=90)
yticks(range(0, nregions), con.region_labels, fontsize=10)
cb=colorbar(shrink=0.8)
cb.set_label('weights', fontsize=14)
show()
```



In [9]:

```
# Initialise the Model

mod = models.ReducedWongWang(a=numpy.array([0.27]), w=numpy.array([1.0]), I_o=numpy.
S = linspace(0, 1, 50).reshape((1, -1, 1))
C = S * 0.0
dS = mod.dfun(S, C)


sim = simulator.Simulator(
    model=mod,
    connectivity=con,
```

```
    coupling=coupling.Linear(a=np.array([0.5 / 50.0])),  # initalise the coupling fu
    integrator=integrators.EulerStochastic(dt=1, noise=noise.Additive(nsig=numpy.arr
    monitors=(monitors.TemporalAverage(period=1),),
    simulation_length=60e3
).configure()

(time, data), = sim.run()
```

In [10]:

```
# Perform simulation.
tic = tm.time()

tavg_time, tavg_data = [], []
for tavg in sim(simulation_length=60000):
    if not tavg is None:
        tavg_time.append(tavg[0][0])
        tavg_data.append(tavg[0][1])

'simulation required %0.3f seconds.' % (tm.time()-tic)
```

Out[10]: 'simulation required 10.567 seconds.'

In [19]:

```
# Normalize time series
tavg_data /= (np.max(tavg_data, 0) - np.min(tavg_data, 0))
tavg_data -= np.mean(tavg_data, 0)

# Make lists numpy.arrays for easier use.
TAVG = np.squeeze(np.array(tavg_data))
TAVG.shape
```

Out[19]: (60000, 76)

In [18]:

```
DATA = mod.p[[0]] * TAVG[:, 0, :] + (1 - mod.p[[0]]) * TAVG[:, 2, :]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-18-5ed8bfd56992> in <module>
----> 1 DATA = mod.p[[0]] * TAVG[:, 0, :] + (1 - mod.p[[0]]) * TAVG[:, 2, :]

AttributeError: 'numpy.ufunc' object has no attribute 'p'
```

In [14]:

```
# Plot time series.
fig1 = plt.figure(figsize=(15,15))
plt.plot(DATA[10000:20000, :] + r_[:nregions], 'k', alpha=0.5)
plt.title('Resting-state time series', fontsize=20)
plt.xlabel('Time [ms]', fontsize=20)
plt.yticks(np.arange(len(con.region_labels)), con.region_labels, fontsize=10)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-14-cf5b223652a3> in <module>
      1 # Plot time series.
      2 fig1 = plt.figure(figsize=(15,15))
----> 3 plt.plot(DATA[10000:20000, :] + r_[:nregions], 'k', alpha=0.5)
      4 plt.title('Resting-state time series', fontsize=20)
      5 plt.xlabel('Time [ms]', fontsize=20)

NameError: name 'DATA' is not defined
```