```
n_legendre_regressors = 7

# Initiate response_fytter object
rfyt = ResponseFytter(
        input_signal=resampled_timecourse,
        sample_rate=deconvolution_frequency,
        oversample_design_matrix=20
        )

# Add events

# Stimulus onset events (with durations)
rfyt.add_event(basis_set = 'legendre',
        n_regressors=n_legendre_regressors,
        event_name=all_event_names[0],
        onset_times = stim_event_list,
        durations=stim_duration_list,
        interval=deconvolution_interval)

# Feedback events (impulse)
        rfyt.add_event(basis_set = 'legendre',
        n_regressors=n_legendre_regressors,
        event_name=all_event_names[1],
        onset_times = fb_event_list[0],
        interval=deconvolution_interval)

rfyt.add_event(basis_set = 'legendre',
        n_regressors=n_legendre_regressors,
        event_name=all_event_names[2],
        onset_times = fb_event_list[1],
        interval=deconvolution_interval)

# Feedback covariates (impulse)
        rfyt.add_event(basis_set = 'legendre',
        n_regressors=n_legendre_regressors,
        event_name=all_event_names[3],
        onset_times = fb_event_list[0],
        covariates=fb_RPEs[0],
        interval=deconvolution_interval)

rfyt.add_event(basis_set = 'legendre',
        n_regressors=n_legendre_regressors,
        event_name=all_event_names[4],
        onset_times = fb_event_list[1],
        covariates=fb_RPEs[1],
        interval=deconvolution_interval)
```

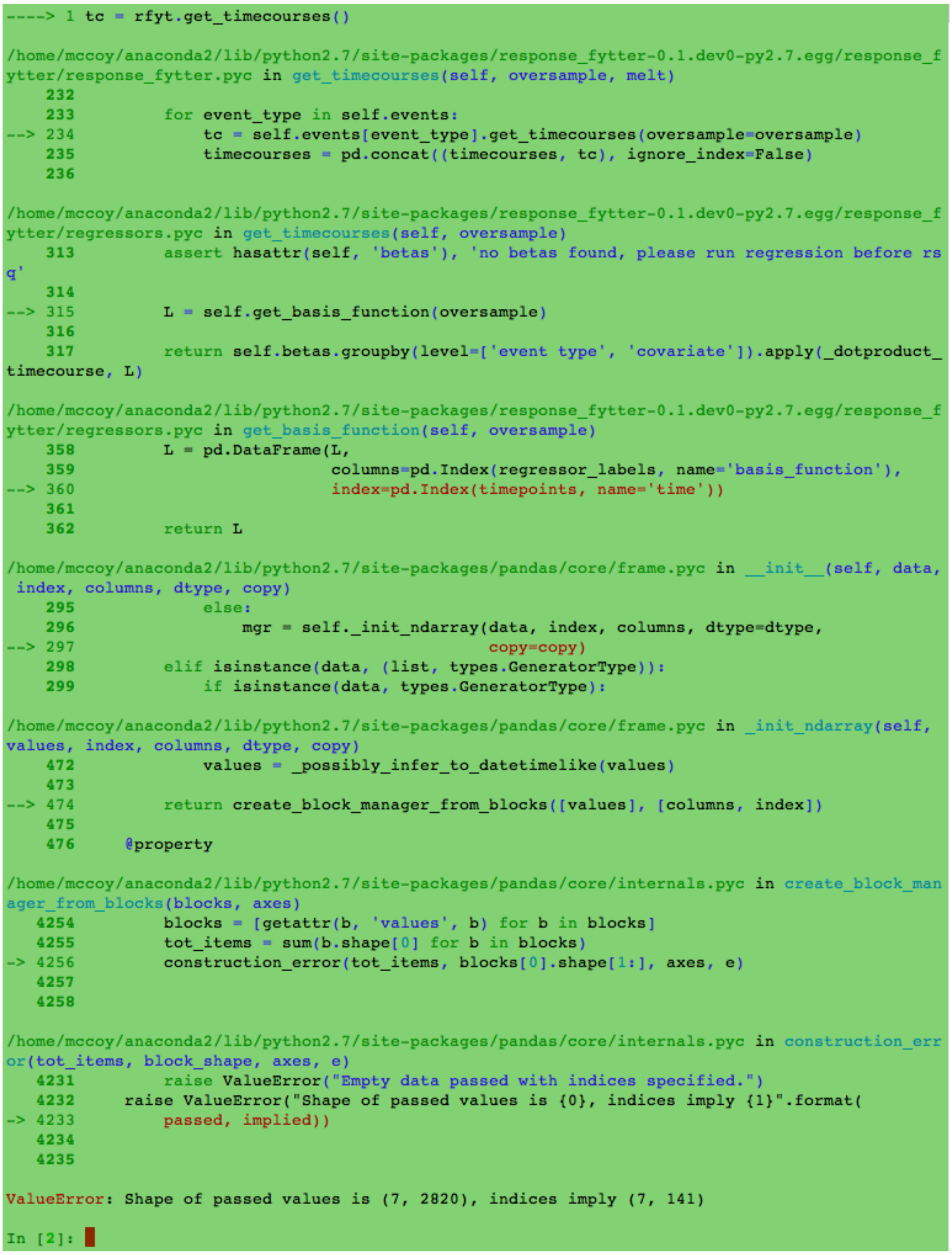
```
# Add confounds
all_nuisances=sp.signal.resample([confounds_df['FramewiseDisplacement'].values,confoun
ds_df['X'].values,confounds_df['Y'].values,confounds_df['Z'].values,confounds_df['RotX'].val
ues,confounds_df['RotY'].values,confounds_df['RotZ'].values,confounds_df['WhiteMatter'].
values,confounds_df['stdDVARS'].values,confounds_df['aCompCor00'].values,confounds_df[
'aCompCor01'].values,confounds_df['aCompCor02'].values,confounds_df['aCompCor03'].val
ues,confounds_df['aCompCor04'].values,confounds_df['aCompCor05'].values],
resampled_timecourse.shape[0], axis = -1)

rfyt.add_confounds(confound=np.array(all_nuisances.T),name='all')

# Perform regression
rfyt.regress()

# Get timecourses
tc = rfyt.get_timecourses()
```

ERROR:



```
----> 1 tc = rfyt.get_timecourses()

/home/mccoy/anaconda2/lib/python2.7/site-packages/response_fytter-0.1.dev0-py2.7.egg/response_f
ytter/response_fytter.pyc in get_timecourses(self, oversample, melt)
    232
    233         for event_type in self.events:
--> 234             tc = self.events[event_type].get_timecourses(oversample=oversample)
    235             timecourses = pd.concat((timecourses, tc), ignore_index=False)
    236

/home/mccoy/anaconda2/lib/python2.7/site-packages/response_fytter-0.1.dev0-py2.7.egg/response_f
ytter/regressors.pyc in get_timecourses(self, oversample)
    313         assert hasattr(self, 'betas'), 'no betas found, please run regression before rs
q'
    314
--> 315         L = self.get_basis_function(oversample)
    316
    317         return self.betas.groupby(level=['event type', 'covariate']).apply(_dotproduct_
timecourse, L)

/home/mccoy/anaconda2/lib/python2.7/site-packages/response_fytter-0.1.dev0-py2.7.egg/response_f
ytter/regressors.pyc in get_basis_function(self, oversample)
    358         L = pd.DataFrame(L,
    359                          columns=pd.Index(regressor_labels, name='basis_function'),
--> 360                          index=pd.Index(timepoints, name='time'))
    361
    362         return L

/home/mccoy/anaconda2/lib/python2.7/site-packages/pandas/core/frame.pyc in __init__(self, data,
 index, columns, dtype, copy)
    295             else:
    296                 mgr = self._init_ndarray(data, index, columns, dtype=dtype,
--> 297                                          copy=copy)
    298         elif isinstance(data, (list, types.GeneratorType)):
    299             if isinstance(data, types.GeneratorType):

/home/mccoy/anaconda2/lib/python2.7/site-packages/pandas/core/frame.pyc in _init_ndarray(self,
values, index, columns, dtype, copy)
    472             values = _possibly_infer_to_datetimelike(values)
    473
--> 474         return create_block_manager_from_blocks([values], [columns, index])
    475
    476     @property

/home/mccoy/anaconda2/lib/python2.7/site-packages/pandas/core/internals.pyc in create_block_man
ager_from_blocks(blocks, axes)
   4254         blocks = [getattr(b, 'values', b) for b in blocks]
   4255         tot_items = sum(b.shape[0] for b in blocks)
-> 4256         construction_error(tot_items, blocks[0].shape[1:], axes, e)
   4257
   4258

/home/mccoy/anaconda2/lib/python2.7/site-packages/pandas/core/internals.pyc in construction_err
or(tot_items, block_shape, axes, e)
   4231         raise ValueError("Empty data passed with indices specified.")
   4232     raise ValueError("Shape of passed values is {0}, indices imply {1}".format(
-> 4233         passed, implied))
   4234
   4235

ValueError: Shape of passed values is (7, 2820), indices imply (7, 141)

In [2]: █
```

```
### Adapted ###

n_timepoints = (deconvolution_interval[1]-deconvolution_interval[0]) / TR

timepoints = np.arange(deconvolution_interval[0],deconvolution_interval[1] + (1./rfyt.sample_rate/rfyt.oversample_design_matrix),1./rfyt.sample_rate / rfyt.oversample_design_matrix)

# Response_fytter currently has this in _create_legendre_basis() function
# x = np.linspace(-1, 1, len(timepoints) * rfyt.oversample_design_matrix, endpoint=True)

# Removing oversample scaling here to make L dataframe have the right number of timepoints
x = np.linspace(-1, 1, len(timepoints), endpoint=True)

L = np.polynomial.legendre.legval(x=x, c=np.eye(n_legendre_regressors)).T

regressor_labels = ['legendre_%d' % poly for poly in np.arange(1, n_legendre_regressors + 1)]

L = pd.DataFrame(L, columns=pd.Index(regressor_labels, name='basis_function'),
                                                index=pd.Index(timepoints, name='time'))

def _dotproduct_timecourse(d, L):
        return L.dot(d.reset_index(level=['event type', 'covariate'], drop=True))

# Remove confounds
rfyt.betas=rfyt.betas.drop('confounds')

tc = rfyt.betas.groupby(level=['event type', 'covariate']).apply(_dotproduct_timecourse, L)
```