Build in functions

Varying effects

Varying intercepts

```
from BI import bi
import numpy as np

# Setup device--------------------------------------------------
m = bi(platform='cpu')

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'reedfrogs.csv'
m.data(data_path, sep=';')
# Manipulate
m.df["tank"] = np.arange(m.df.shape[0])

# Define model -------------------------------------------------
def model(tank, surv, density):
    alpha = m.effects.varying_intercept(group=tank,group_name = 'tank')
    m.dist.binomial(total_count = density, logits = alpha, obs=surv)

# Run sampler --------------------------------------------------
m.fit(model)

# Diagnostic ---------------------------------------------------
m.summary()
```

Varying slopes

```
```

Varying intercepts and slopes

```
from BI import bi
# Setup device--------------------------------------------------
m = bi(platform='cpu')

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'Sim data multivariatenormal.csv'
m.data(data_path, sep=',')

# Define model -------------------------------------------------
def model(cafe, wait, N_cafes, afternoon):
    a = m.dist.normal(5, 2,  name = 'a')
    b = m.dist.normal(-1, 0.5, name = 'b')
    sigma = m.dist.exponential( 1,  name = 'sigma')

    varying_intercept, varying_slope = m.effects.varying_effects(
        N_group = N_cafes,
        group = cafe,
        global_intercept= a,
        global_slope= b,
        group_name = 'cafe')


    mu = varying_intercept + varying_slope* afternoon
    m.dist.normal(mu, sigma, obs=wait)

# Run sampler --------------------------------------------------
m.fit(model)
```

Gaussian processes

Squared Exponential Kernel

```
from BI import bi
# Setup device--------------------------------------------------
m = bi(platform='cpu')
m.gaussian.kernel_sq_exp
```

Periodic Kernel

```
from BI import bi
# Setup device--------------------------------------------------
m = bi(platform='cpu')
m.gaussian.kernel_periodic
```

Locally Periodic Kernel

```
from BI import bi
# Setup device--------------------------------------------------
m = bi(platform='cpu')
m.gaussian.kernel_periodic_local
```

Networks effects

Sender receiver

```
sr =  m.net.sender_receiver(focal_individual_predictors,target_individual_predictors)
```

Dyadic

```
m.net.dyadic_effect(dyadic_predictors)
```

Block model

```
m.net.block_model(Merica: vector[integer],3)
```

Network metrics

Degree (undirected, out-degree, in-degree)

```
m.net.degree(adj_matrix_jax)
m.net.indegree(adj_matrix_jax)
m.net.outdegree(adj_matrix_jax)
```

Strength (undirected, out-strength , in-strength)

```
m.net.strength(adj_matrix_jax)
m.net.instrength(adj_matrix_jax)
m.net.outstrength(adj_matrix_jax)
```

Eigenvector

```
m.net.eigen(adj_matrix_jax)
```

Clustering coefficient

```
m.net.eigen(adj_matrix_jax)
```

Density

```
m.net.eigen(adj_matrix_jax)
```

Geodesic_distance

```
m.net.eigen(adj_matrix_jax)
```

Diameter

```
m.net.eigen(adj_matrix_jax)
```