Loading in packages needed for data analysis

```
import arviz as az
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pymc3 as pm
import seaborn as sns
import theano.tensor as tt
import matplotlib.pyplot as plt
import matplotlib.font_manager as font_manager
from scipy import stats
import seaborn as sns
import statsmodels.api as sm
from statsmodels.sandbox.regression.predstd import wls_prediction_std
import bambi as bmb


RANDOM SEED = 12345
```

Defining model formulas for Bayesian Models

```
anx_formula = "W2_TS_ANX_t ~ W2_LV_dichotomous + W2_HC_dichotomous +
W2 BE dichotomous + W2_Boston + W2_SanDiego + W2 SES zscore + W2 lat orient
+ W2_anglo_orient + IMG status + male"
dep_formula = "W2_TS_D_t ~ W2_LV_dichotomous + W2 HC dichotomous +
W2 BE dichotomous + W2_Boston + W2_SanDiego + W2 SES zscore + W2 lat orient
+ W2_anglo_orient + IMG status + male"
ang_formula = "W2_TS_ANG_t ~ W2_LV_dichotomous + W2 HC dichotomous +
W2 BE dichotomous + W2_Boston + W2_SanDiego + W2_SES_zscore + W2_lat_orient
+ W2_anglo_orient + IMG status + male"
dis_formula = "W2_TS_DIS_t ~ W2 LV dichotomous + W2 HC dichotomous +
W2 BE dichotomous + W2_Boston + W2_SanDiego + W2 SES zscore + W2 lat orient
+ W2_anglo_orient + IMG status + male"
```

Running models in Bambi

Anxiety

Weakly Informed

Model syntax and observing model priors

```
anx_weak_priors = {
        "Intercept": bmb.Prior("Normal", mu=50, sigma=11,),
        "W2_LV_dichotomous": bmb.Prior("Normal", mu=0, sigma=20,),
        "W2_HC_dichotomous": bmb.Prior("Normal", mu=0, sigma=20),
        "W2_BE_dichotomous": bmb.Prior("Normal", mu=0, sigma=20),}
```

```
anx uninform = bmb.Model(anx_formula,  w2_d_anx, priors=anx_weak_priors,
family = 't')
anx_uninform.build()
anx_un_fitted = anx_uninform.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
print (anx_uninform)
anx_uninform.graph(name="anx_uninform")
```

Trace plots

```
az.plot_trace(anx_un_fitted, compact=False, combined=False)
```

Autocorrelation plots

```
az.plot_autocorr(anx_un_fitted, combined = True)
```

Model summary

```
ax sum = az.summary(anx_un_fitted,  hdi_prob=.95,round_to=4)


ax_sum.to_csv("ax_results.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(anx_un_fitted, hdi_prob=.95, kind="hist", rope=(-
w2_d_anx.W2_TS_ANX_t.std()*.1, w2_d_anx.W2_TS_ANX_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Highly Informed

```
anx_priors = {
       "Intercept": bmb.Prior("Normal", mu=42.5, sigma=2.1),
       "W2_LV_dichotomous": bmb.Prior("Normal", mu=4.5, sigma=.7,),
       "W2_HC_dichotomous": bmb.Prior("Normal", mu=2.6, sigma=.9,),
       "W2_BE_dichotomous": bmb.Prior("Normal", mu=2.3, sigma=.8),}


anx int inf = bmb.Model(anx_formula,  w2_d_anx, priors=anx_priors, family =
"t")
anx_int_inf.build()
```

```
anx_fitted_int = anx_int_inf.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
anx_int_inf.graph(name="anx_int_inf")
```

Trace plots

```
az.plot_trace(anx_fitted_int, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(anx_fitted_int, combined = True)
```

Model summary

```
ax informed = az.summary(anx_fitted_int,  hdi_prob=.95,round_to=5)


ax_informed.to_csv("ax_informed_results.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(anx_fitted_int, hdi_prob=.95, kind="hist", rope=(-
w2_d_anx.W2_TS_ANX_t.std()*.1, w2_d_anx.W2_TS_ANX_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Anger

Weakly Informed

Model syntax and observing model priors

```
ang_weak_priors = {
        "Intercept": bmb.Prior("Normal", mu=48, sigma=50,),
        "W2_LV_dichotomous": bmb.Prior("Normal", mu=0, sigma=20,),
        "W2 HC dichotomous": bmb.Prior("Normal", mu=0, sigma=20),
        "W2_BE_dichotomous": bmb.Prior("Normal", mu=0, sigma=20),}


ang_uninform = bmb.Model(ang_formula,  w2_d_ang, priors=ang_weak_priors,
family= "t")
ang_uninform.build()
ang_un_fitted = ang_uninform.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
```

```
print (ang_uninform)
ang_uninform.graph(name="ang_uninform")
```

Trace plots

```
az.plot_trace(ang_un_fitted, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(ang_un_fitted, combined = True)
```

Model summary

```
ang_sum = az.summary(ang_un_fitted,  hdi_prob=.95,round_to=4)


ang_sum.to_csv("ang_sum.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(ang_un_fitted, hdi_prob=.95, kind="hist", rope=(-
w2_d_ang.W2_TS_ANG_t.std()*.1, w2_d_ang.W2_TS_ANG_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Highly Informed

```
ang_priors = {
       "Intercept": bmb.Prior("Normal", mu=41.2, sigma=2.1,),
       "W2_LV_dichotomous": bmb.Prior("Normal", mu=3.3, sigma=.7,),
       "W2_HC_dichotomous": bmb.Prior("Normal", mu=2.8, sigma=.9,),
       "W2_BE_dichotomous": bmb.Prior("Normal", mu=2.1, sigma=.8),}



ang_int_inf = bmb.Model(ang_formula,  w2_d_ang, priors=ang_priors, family =
"t")
ang_int_inf.build()
ang_fitted_int = ang_int_inf.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
ang_int_inf.graph(name="ang_int_inf")
```

Trace plots

```
az.plot_trace(ang_fitted_int, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(ang_fitted_int, combined = True)
```

Model summary

```
ang_informed_sum = az.summary(ang_fitted_int,  hdi_prob=.95,round_to=4)


ang_informed_sum.to_csv("ang_informed_sum.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(ang_fitted_int, hdi_prob=.95, kind="hist", rope=(-
w2_d_ang.W2_TS_ANG_t.std()*.1, w2_d_ang.W2_TS_ANG_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Depression

Weakly informed

Model syntax and observing model priors

```
dep_weak_priors = {
       "Intercept": bmb.Prior("Normal", mu=49, sigma=11,),
       "W2 LV dichotomous": bmb.Prior("Normal", mu=0, sigma=20,),
       "W2 HC dichotomous": bmb.Prior("Normal", mu=0, sigma=20,),
       "W2 BE dichotomous": bmb.Prior("Normal", mu=0, sigma=20),}


dep_uninform = bmb.Model(dep_formula,  w2_d_dep, priors = dep_weak_priors,
family= "t")
dep_uninform.build()
dep_un_fitted = dep_uninform.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
print (dep_uninform)
dep_uninform.graph(name="dep_uninform")
```

Trace plots

```
az.plot_trace(dep_un_fitted, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(dep_un_fitted, combined = True)
```

Model summary

```
dep_sum = az.summary(dep_un_fitted,  hdi_prob=.95,round_to=4)


dep_sum.to_csv("dep_sum.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(dep_un_fitted, hdi_prob=.95, kind="hist", rope=(-
w2_d_dep.W2_TS_D_t.std()*.1, w2_d_dep.W2_TS_D_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Informed

Model syntax and observing model priors

```
dep_priors = {
       "Intercept": bmb.Prior("Normal", mu=41.2, sigma=2.1),
       "W2_LV_dichotomous": bmb.Prior("Normal", mu=3.3, sigma=.7,),
       "W2 HC dichotomous": bmb.Prior("Normal", mu=2.8, sigma=.9,),
       "W2 BE dichotomous": bmb.Prior("Normal", mu=2.1, sigma=.8),}



dep_int_inf = bmb.Model(dep_formula,  w2_d_dep, priors=dep_priors, family=
"t")
dep_int_inf.build()
dep_fitted_int = dep_int_inf.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
dep_int_inf.graph(name="dis_int_inf")
```

Trace plots

```
az.plot_trace(dep_fitted_int, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(dep_fitted_int, combined = True)
```

Model summary

```
dep_informed_sum = az.summary(dep_fitted_int,  hdi_prob=.95,round_to=4)
```

```
dep_informed_sum
```

```
dep_informed_sum.to_csv("dep_informed_sum.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(dep_fitted_int, hdi_prob=.95, kind="hist", rope=(-
w2_d_dep.W2_TS_D_t.std()*.1, w2_d_dep.W2_TS_D_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Dissociation

Weakly informed

Model syntax and observing model priors

```
dis_weak_priors = {
       "Intercept": bmb.Prior("Normal", mu=51, sigma=12,),
       "W2_LV_dichotomous": bmb.Prior("Normal", mu=0, sigma=20,),
       "W2_HC_dichotomous": bmb.Prior("Normal", mu=0, sigma=20,),
       "W2 BE dichotomous": bmb.Prior("Normal", mu=0, sigma=20),}
```

```
dis uninform = bmb.Model(dis_formula,  w2_d_dis, priors=dis_weak_priors,
family = "t")
dis_uninform.build()
dis_un_fitted = dis_uninform.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED, target_accept = .87)
print (dis_uninform)
dis_uninform.graph(name="dis_uninform")
```

Trace plots to observe model convergence

```
az.plot_trace(dis_un_fitted, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(dis_un_fitted, combined = True)
```

Model summary

```
dis sum = az.summary(dis_un_fitted,  hdi_prob=.95,round_to=4)


dis_sum.to_csv("dis_sum.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(dis_un_fitted, hdi_prob=.95, kind="hist", rope=(-
w2_d_dis.W2_TS_DIS_t.std()*.1, w2_d_dis.W2_TS_DIS_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```

Informed priors

Model syntax and observing model priors

```
dis_priors = {
       "Intercept": bmb.Prior("Normal", mu=44, sigma=2.3,),
       "W2_LV_dichotomous": bmb.Prior("Normal", mu=2.5, sigma=.8,),
       "W2 HC dichotomous": bmb.Prior("Normal", mu=3.4, sigma=.9),
       "W2 BE dichotomous": bmb.Prior("Normal", mu=1.3, sigma=.9),}



dis_int_inf = bmb.Model(dis_formula,  w2_d_dis, priors=dis_priors, family =
"t")
dis_int_inf.build()
dis_fitted_int = dis_int_inf.fit(tune=2000, draws=10000, init="adapt_diag",
random_seed=RANDOM_SEED)
print(dis_int_inf)
dis_int_inf.graph(name="dis_int_inf")
```

Trace plots

```
az.plot_trace(dis_fitted_int, compact=True)
```

Autocorrelation plots

```
az.plot_autocorr(dis_fitted_int, combined = True)
```

Summary of informed model

```
dis_informed_sum = az.summary(dis_fitted_int,  hdi_prob=.95,round_to=4)
```

```
dis_informed_sum.to_csv("dis_informed_sum.csv")
```

Posterior distributions and ROPE

```
az.plot_posterior(dis_fitted_int, hdi_prob=.95, kind="hist", rope=(-
w2_d_dis.W2_TS_DIS_t.std()*.1, w2_d_dis.W2_TS_DIS_t.std()*.1),
point_estimate='mean', backend="matplotlib")
```