Zenodo. If you don't want to run the models yourself, you can download them on zenodo in the .netcdf file format with the az.from_netcdf('...').

Note: the DAG used to build all the models in this notebook is described in the manuscript submitted to EJT.

In [16]:
```
# versions

print('\n'.join(f'{m.__name__}=={m.__version__}' for m in globals().values() if
```

```
re==2.2.1
pandas==1.5.3
numpy==1.24.2
pymc==5.1.2
bambi==0.10.0
statsmodels.api==0.13.5
seaborn==0.12.2
arviz==0.15.1
```

In [2]:
```
SEED = 201288
```

In [128…
```
## plotting functions

# function for getting the probabilities

def pordlog(a):

    # transform back to cumulative probabilites
    pa = expit(a)
    p_cum = np.concatenate(([0.], pa, [1.]))

    # get the intervals instead of the cumulative probs
    return p_cum[1:] - p_cum[:-1]

# function for plotting the posterior predictive proportions of ordinal scores

def ordinal_plot(nscores, df, title, labels,axis):
    colors = ['r','b','grey','g','c']

    for i in range(nscores):


        sns.scatterplot(x=new_hits,
                        y=df.iloc[i],
                        alpha = 0.5,
                        color = colors[i],
                        label = labels[i],
                        ax=ax[axis])
    ax[axis].legend()
    ax[axis].set_title(title)


# function for creating a stackplot of the proportions

def plot_probabilities2(ncuts, trace,cuts,name_lin,title,labels,axis,ylim,xrange
    logit = cuts - trace.predictions[name_lin]
    logit_mean = logit.mean(dim = ['chain','draw'])
    probabilities = np.array([pordlog(logit_mean[:,i].values) for i in range(len
```