Student Name: Guangyu Lin

Collaboration Statement:

Total hours spent: a week

I discussed ideas with these individuals:

• Mingyang Wu

• TODO

• ...

I consulted the following resources:

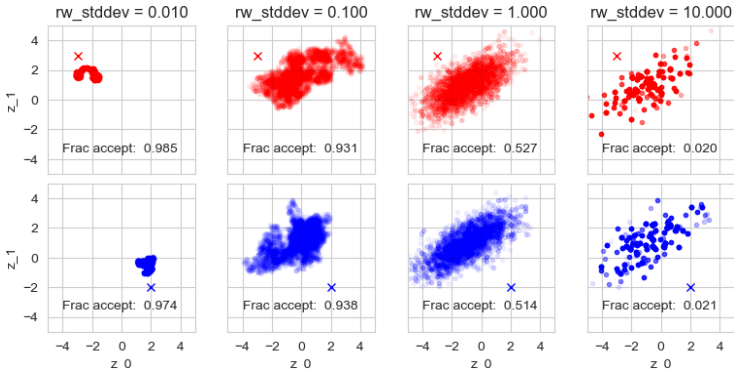
• office hour with Professor and TA

• textbook

• ...

By submitting this assignment, I affirm this is my own original work that abides by the course collaboration policy.

1a

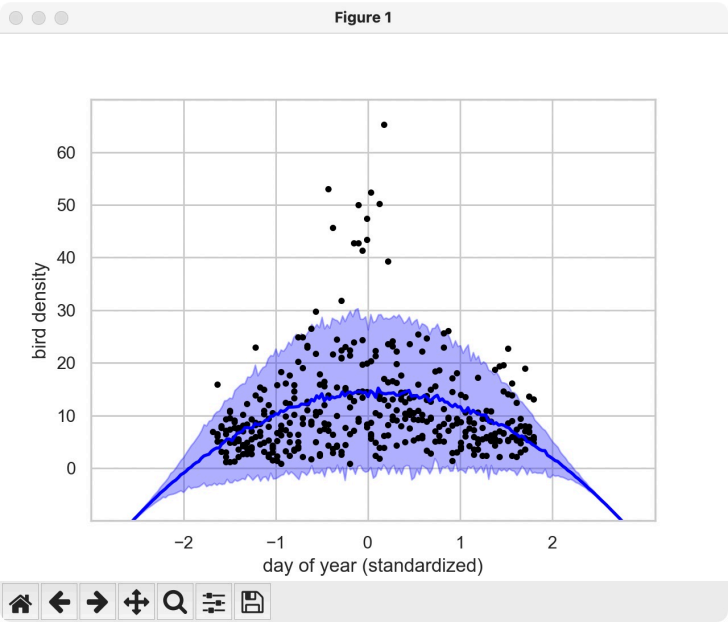


1b

No, we are not confident towards the MCMC chain has converged. There are two reasons. First, based on 1a, we know the graph is converged better when accept rate is 0.5 than 0.9. Therefore, we can't

guarantee when accept rate is 0.8, the MCMC chain has converged. Second, it only tried one trial and we can't guarantee it stated with a good position or not. If we can try more trials with different start points, we may have more confident to say it is comverged.

2a



2b

Order	test score
0	-4.513134
2	-4.218459

2c

```
def calc_score(list_of_z_D, phi_RM, t_R):
    ''' Calculate per-example score averaged over provided test set of size R

    Args
    ----
    list_of_z_D : list of ndarray
```

```
        List of samples of parameters, assumed to be from target posterior
    phi_RM : 2D array, shape (R, M)
        Feature vectors for each of the examples in test set of size R
    t_R : 1D array, shape (R,)
        Output values for each of the examples in test set of size R

    Returns
    -------
    score : float
        Per-example log pdf of all t values in test set
        using Monte-Carlo approximation to marginal likelihood
    '''
    S = len(list_of_z_D)
```

```
    log_likelihoods = []
    for ss in range(S):
        z_ss_D = list_of_z_D[ss]
        mean_R, stddev_R =
unpack_mean_N_and_stddev_N(z_ss_
D, phi_RM)
        log_likelihoods.append(

 scipy.stats.norm.logpdf(t_R,
loc=mean_R, scale=stddev_R)
        )
        # Compute score formula
for ss-th sample (see
instructions)
        # Hint: Use
unpack_mean_N_and_stddev_N
    # TODO aggregate across all
S samples
    # Hint: use
scipy.special.logsumexp to be
numerically stable
    log_likelihoods =
np.asarray(log_likelihoods)
```

```
32      score =
    np.mean(scipy.special.logsumexp(
    log_likelihoods, axis = 0) -
    np.log(S))
33      return score # TODO FIXME
```