Florida Bay Assessment Model Update Notes

Proposed Update: Add surface water temperature dependent evaporation amplification.

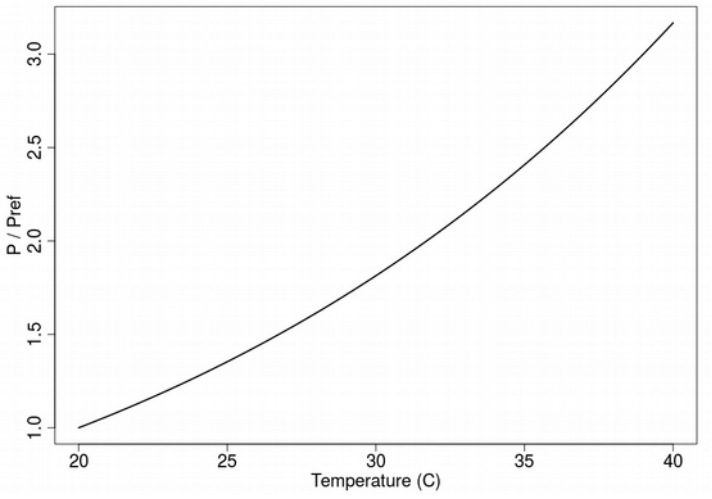
It has been noted that during the Florida Bay hypersalinity event in July 2015, BAM does not estimate the peak hypersalinities (above 50 ppt) well. In part, this is likely due to the large spatial scales and basin-wide averages inherent in BAM, whereas the observations are point measurements. Another potential issue is that ET is represented by a single timeseries derived from USGS GOES estimates in the southern EDEN domain (marsh).

In an attempt to refine BAM salinity estimates, the proposed update applies a temperature-based ET amplification, as described below.

The idea is that observed water surface temperature can be used to estimate a ratio of increased equilibrium vapor pressure, and thus evaporation. This increased evaporation rate is then applied to specified basins. The vapor pressure change is quantified by the Clausius-Clapeyron relation:

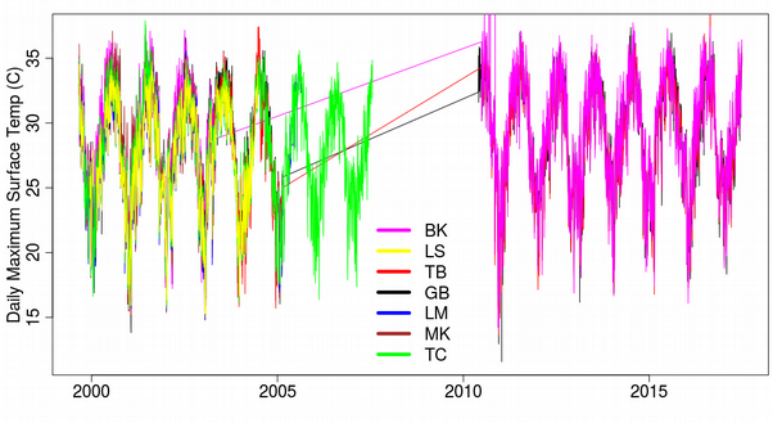
$$\ln\left(\frac{P}{P_{ref}}\right) = \frac{\Delta H}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)$$

where P is the vapor pressure, T temperature in Kelvins, ΔH the change in specific enthalpy (specific latent heat) and R the universal gas constant.



Water vapor pressure ratio based on a reference temperature of 20 C.

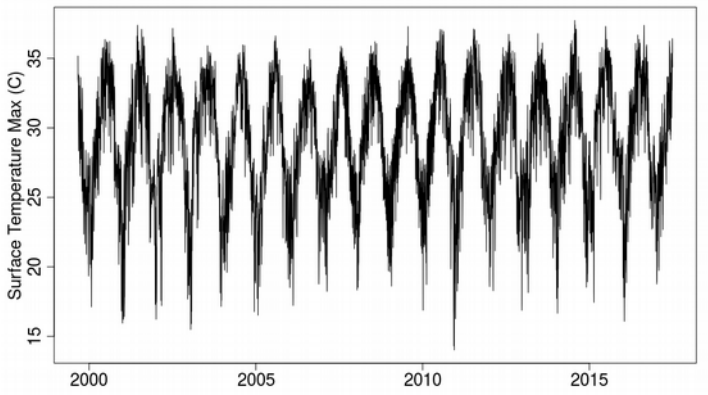
Temperature Data



Coastal water surface daily maximum temperature data available in Data4EVER over the BAM period-of-record.

Surface temperatures are largely coherent and similar in magnitude across basins.

The large gap will need data reconstruction.



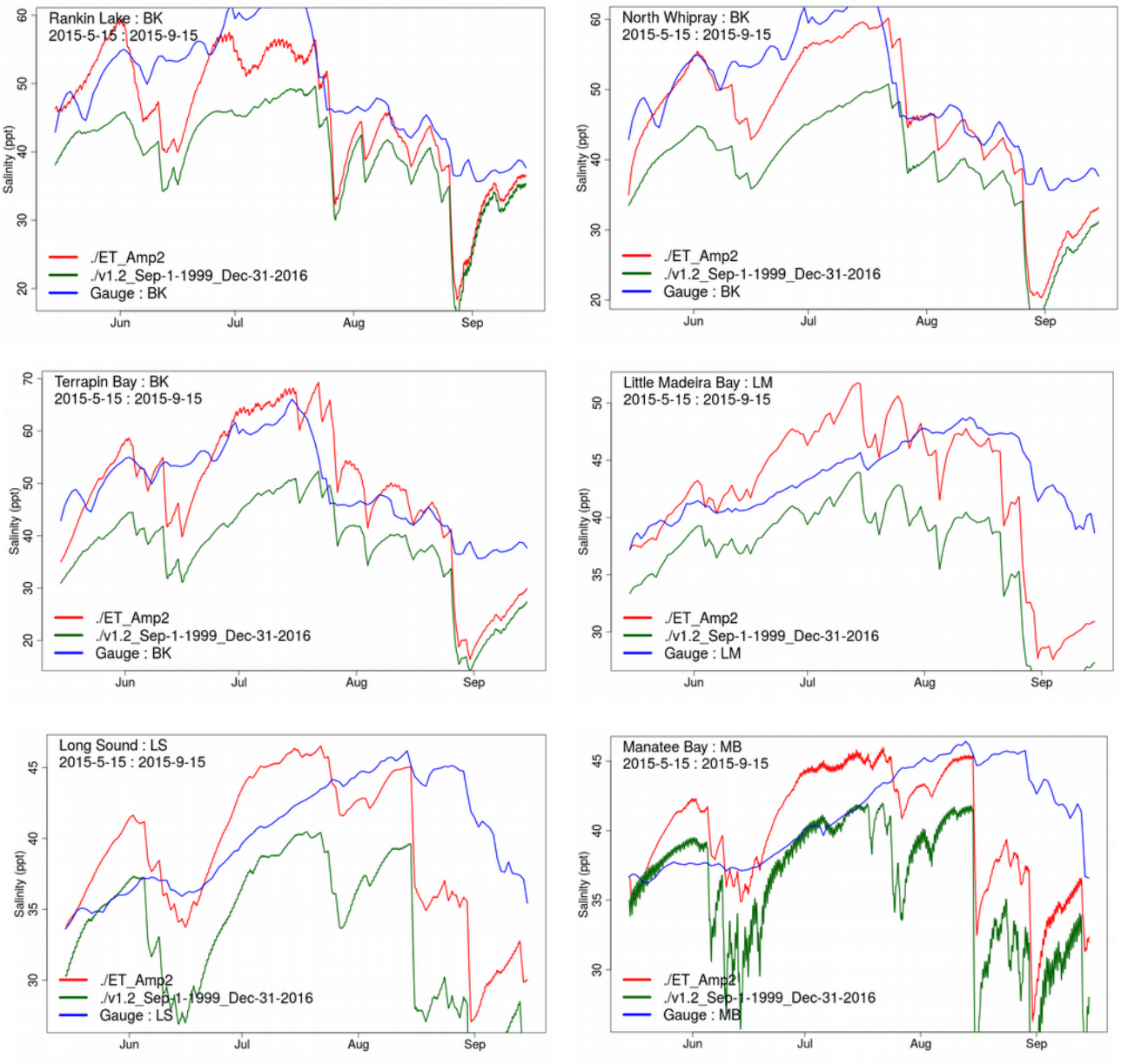
Maximum daily water surface temperature data and reconstruction for BAM.

BK data was used where available. Where GB was available in the BK gaps, linear regression was used to estimate BK data. Where TC was available in the remaining gaps, linear regression was used to estimate BK data. The remaining gaps were filled with a Gaussian-based empirical resampling for equivalent year-days based on existing data.

BAM was modified with the addition of `GetTemperature()` and `VaporPressureRatio()` methods in the `model` object, and the `GetTemperatureData()` method in the `init` module. The temperature timeseries was added in the `data/Temperature` directory. Command line options to control these inputs are:

```
-st --temperature  Temperature data file
-ea --ET_Amplify   Apply basin ET amplification from vapor pressure ratio
-rt --reference_temperature  Vapor pressure reference temperature
```

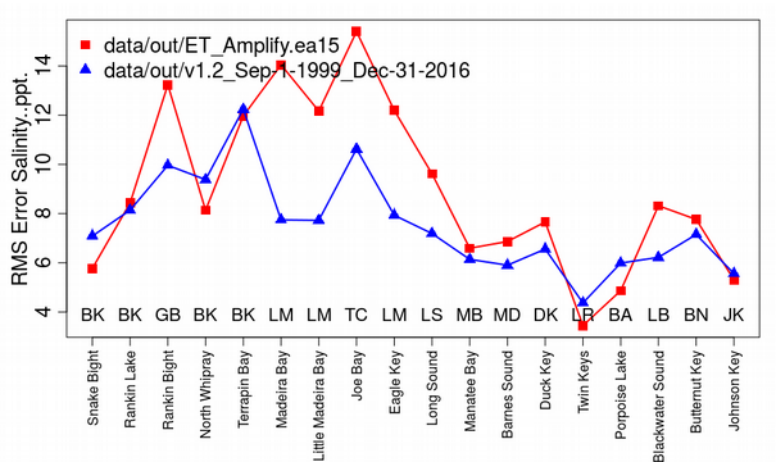
The ET amplification is only applied to basins in the basins parameter file (`-bp --basinParameter`) with the `ET Amplify` field set `True` .



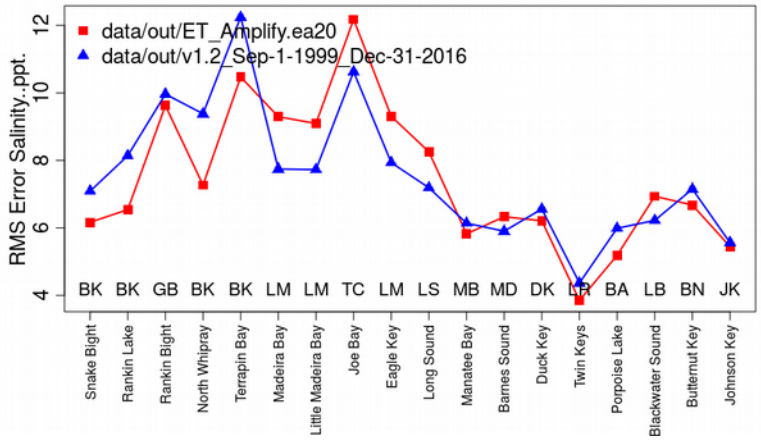
Comparisons of model results with ET amplification to version 1.2 without ET amplification.

Red : With ET amplification at reference temperature of 20 C.
Green : Without ET amplification
Blue : Gauge point-measurements

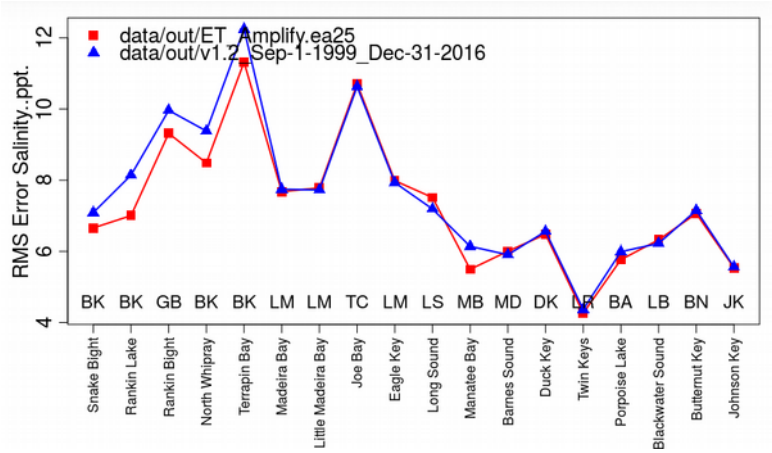
RMS Error (ppt) over the period 1999-9-1 to 2016-12-31 at basins in proximity to gauges.
BAM was run with ET Ampify for basins: Terrapin Bay, North Whipray, Rankin Lake, Snake Bight.



a) Reference temperature 15 C.



b) Reference temperature 20 C.



c) Reference temperature 25 C.

The reference temperature (`-rt`) option has a default value of 25.

Code Excerpts:

```
    #---------------------------------------------------------------
    #
    #---------------------------------------------------------------
    def GetTemperature( self, key ):
        '''Get water temperature for basin, but only if the ET Amplify
           option is specified (-ea) and the basin ET Amplify field
           is True in Basin_Parameters.csv'''

        if self.args.DEBUG_ALL :
            print( '\n-> GetTemperature', flush = True )

        if not self.args.ET_amplify :
            return

        temperature = self.temperature_data[ key ]

        for Basin in self.Basins.values() :
            if Basin.boundary_basin :
                continue

            if Basin.ET_amplify :
                Basin.temperature = temperature

    #---------------------------------------------------------------
    #
    #---------------------------------------------------------------
    def GetET( self, key ):
        '''Subtract ET volume from basin'''

        if self.args.DEBUG_ALL :
            print( '\n-> GetET', flush = True )

        if self.args.noET :
            return

        et_mm_day = self.et_data[ key ]

        for Basin in self.Basins.values() :
            if Basin.boundary_basin :
                continue

            kinetic_ET_factor = 1

            if self.args.ET_amplify :
                if Basin.temperature :
                    kinetic_ET_factor = self.VaporPressureRatio( \
                                            Basin.temperature )

            et_volume_day = ( et_mm_day / 1000 ) * Basin.area * \
                            self.args.ET_scale * kinetic_ET_factor

            et_volume_t   = et_volume_day / self.timestep_per_day

            Basin.evaporation = et_volume_t

            Basin.water_volume -= et_volume_t
```

```
#-------------------------------------------------------------------
#
#-------------------------------------------------------------------
def VaporPressureRatio( self, temperature ):
    '''Return relative vapor pressure to amplify ET'''

    if self.args.DEBUG_ALL :
        print( '\n-> VaporPressureRatio', flush = True )

    if not self.args.ET_amplify :
        return 1

    dH = 44000 # enthalpy of vaporization J/mol @ 300 K
    R  = 8.314 # universal gas constant   J/(mol K)
    ref_temperature = self.args.reference_temperature

    # Clausius-Clapeyron relation:
    # ln( P2/P1 ) = ( dH / R ) * (1/T2 - 1/T1)
    P_Pref = exp( (dH/R) * ( 1/(ref_temperature + 273.15) -
                             1/(temperature     + 273.15) ) )

    # Limit ratio's less than 1 to 1 (when temp < ref_temp )
    return max( 1, P_Pref )
```