Summary of data processing for Patuxent River Estuary

Marcus W. Beck, Rebecca Murphy

March 18, 2015

The following describes additional processing of raw station data for the Patuxent River Estuary. Briefs descriptions of each step are provided.

The following raw data files were imported:

1. `PAX_TRIB_CHLAandSALINITY_85to14.csv`: chlorophyll and salinity data for all stations in the Patuxent River from 1985 to 2014 (from R. Murphy)
2. `PAX_station_info.csv`: metadata for each station including lat/lon, salinity zone, etc. (from R. Murphy)
3. `cb5_1_w.csv`: Salinity data for all depth zones at station CB5.1W at the outflow of the Patuxent River Estuary, used to calculate fraction of freshwater

The data were first imported into R.

```
# code for processing raw data, see email from R. Murphy on 3/13/15
# created March 2015, M. Beck

## packages to use
# this is just to load dplyr, ggplot2
devtools::load_all('M:/docs/tidal_comp/TidalComp')

## import

# meta
pax_meta <- system.file('PAX_station_info.csv', package = 'TidalComp')
pax_meta <- read.csv(pax_meta, header = TRUE,
  stringsAsFactors = FALSE)
```

```
# data
pax_data <- system.file('PAX_TRIB_CHLAandSALINITY_85to14.csv',
  package = 'TidalComp')
pax_data <- read.csv(pax_data, header = TRUE,
  stringsAsFactors = FALSE)

# reorder STATION variable along trib axis
stats <- c('TF1.0', 'TF1.3', 'TF1.4', 'TF1.5', 'TF1.6', 'TF1.7',
  'RET1.1', 'LE1.1', 'LE1.2', 'LE1.3', 'LE1.4')
pax_data$STATION <- factor(pax_data$STATION, level = stats)

# most seaward station for fraction of freshwater
ffw_data <- system.file('cb5_1_w.csv', package = 'TidalComp')
ffw_data <- read.csv(ffw_data, header = TRUE,
  stringsAsFactors = FALSE)
```

Salinity data (ppt) at the most seaward station (CB5.1W) were vertically integrated for each unique date. The integration function averaged all salinity values after interpolating from the surface to the maximum depth. Salinity values at the most shallow and deepest sampling depth were repeated for zero depth and maximum depths, respectively, to bound the interpolations within the range of the data. A moving window average was also used to aggregate salinity values for all preceding two months for each observation. This approach was used to account for lags in water exchange near the estuary mouth with the mainstem based on previously estimated residence times.

```
##
# process CB5.1W data for fraction of freshwater

# vertical integration by date, station
int_fun <- function(TOTAL_DEPTH, DEPTH, VALUE){

  if(length(na.omit(VALUE)) < 2 ) return(na.omit(VALUE))

  # setup for interpolation
  max_depths <- mean(unique(TOTAL_DEPTH), na.rm = TRUE)
  depths <- c(0, DEPTH, max_depths)
  vals <- c(VALUE[1], VALUE, VALUE[length(VALUE)])

  # only interpolate if > 1 salinity value
  out <- mean(approx(depths, vals)$y)

  return(out)

  }
```

```
# for filter
wins <- 2# 2 months
wins <- rep(1, length = wins)/wins

# process
ffw_data <- select(ffw_data, SAMPLE_DATE, DEPTH, VALUE, TOTAL_DEPTH) %>%
  group_by(SAMPLE_DATE) %>%
  summarize(sal_ref = int_fun(TOTAL_DEPTH, DEPTH, VALUE)) %>%
  na.omit %>%
  mutate(date = as.Date(SAMPLE_DATE, '%m/%d/%Y')) %>%
  select(date, sal_ref) %>%
  arrange(date) %>%
  mutate(sal_ref = stats::filter(sal_ref, wins, sides = 1))
```

A similar process for vertically-integrating salinity across depth values was used for the remaining station data.

```
##
# get vertically integrated salinity as before

# process
# note that there are no 'PROBLEM' values, lab and method do not change
sal_tmp <- filter(pax_data, PARAMETER == 'SALINITY') %>%
  mutate(date = as.Date(date, format = '%m/%d/%Y')) %>%
  group_by(date, STATION) %>%
  summarize(sal = int_fun(TOTAL_DEPTH, DEPTH, AvgValue))
```

Chlorophyll values at each station were retained only for surface samples and no 'problem' codes. Chlorophyll were also transformed by the natural-log.

```
##
# get only surface estimates for chlorophyll
# remove those w/ problem codes
chl_tmp <- filter(pax_data,
  PARAMETER == 'CHLA' & LAYER == 'S'& PROBLEM == ''
  ) %>%
  mutate(lnchla = log(AvgValue)) %>%
  mutate(date = as.Date(date, format = '%m/%d/%Y')) %>%
  select(date, STATION, lnchla)
```

The reference salinity data at CB5.1W was merged with the salinity data at each date and station to estimate fraction of freshwater. Sample dates between the reference station and

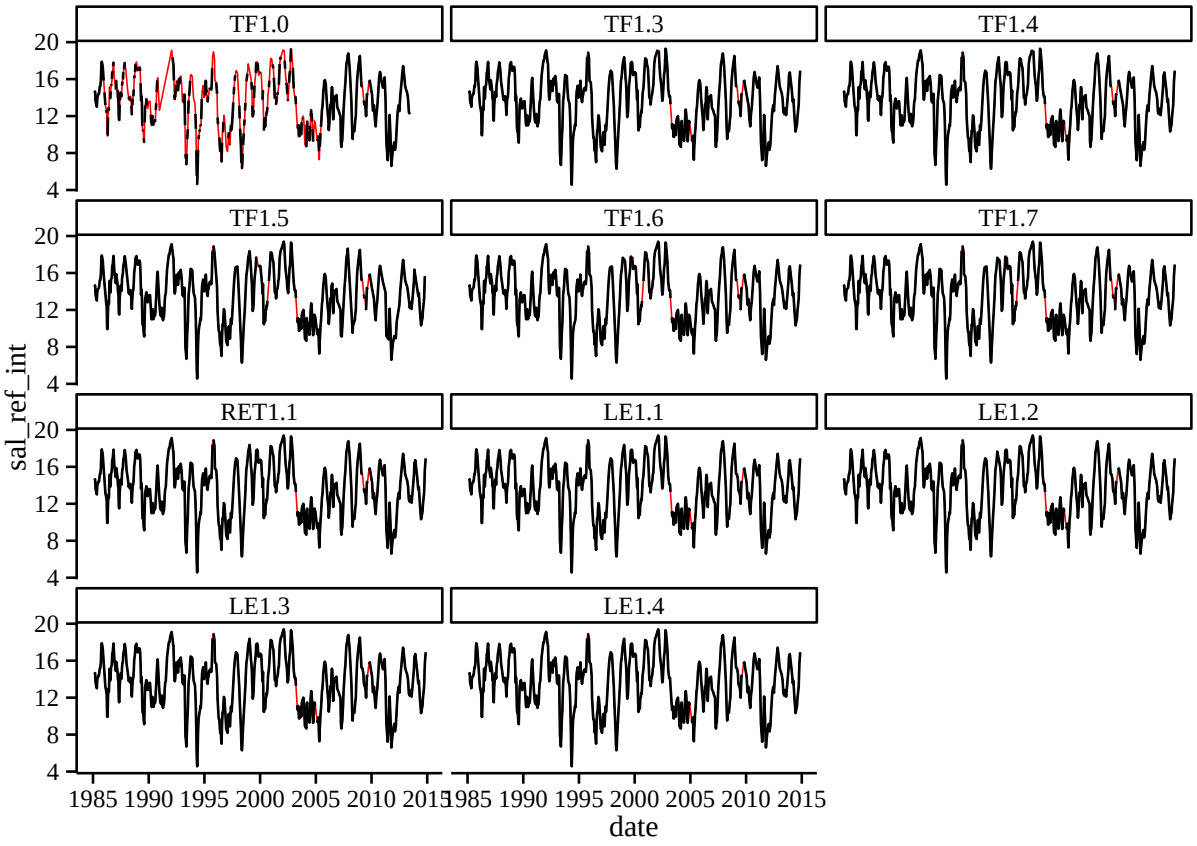
all remaining stations were similar, although in some cases the dates did not match exactly. Again, linear interpolation was used to fill in missing observation dates for salinity at the reference site to match with observation dates at the remaining stations. The corresponding plot shows dates at each station that required interpolation at the reference station.

```
##
# ref salinity dates for CB5.1W need to match sample dates from rest of trib
# CB5.1W data are merged with all vertically-integrated salinity data
# missing dates for CB5.1W are interpolated to get matches

int_fun <-  function(date, sal_ref){
  interps <- approx(x = date, y  = sal_ref, xout = date)$y
  sal_ref[is.na(sal_ref)] <- interps[is.na(sal_ref)]
  return(sal_ref)
}

sal_mrg <- full_join(sal_tmp, ffw_data, by = 'date') %>% # merge
  ungroup %>%
  arrange(STATION, date) %>% # sort by station, date
  group_by(STATION) %>%
  mutate(sal_ref_int = int_fun(date, sal_ref))

# plot the interpolated values to see if they're ok
# red lines are values that were interpolated
ggplot(sal_mrg, aes(x = date, y = sal_ref_int, group = STATION)) +
  geom_line(colour = 'red', size = 0.3) +
  geom_line(aes(y = sal_ref)) +
  facet_wrap(~STATION, ncol = 3) +
  theme_classic()
```



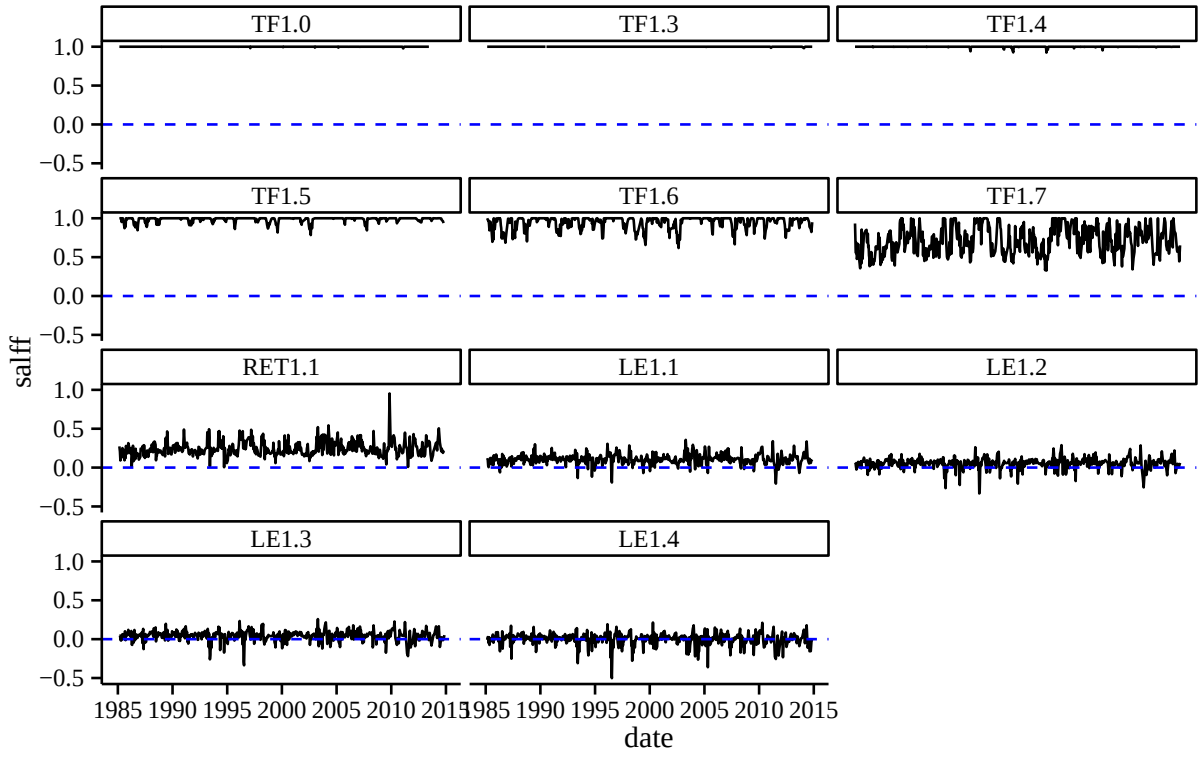
The fraction of freshwater data were merged with the chlorophyll data. All unique data were retained.

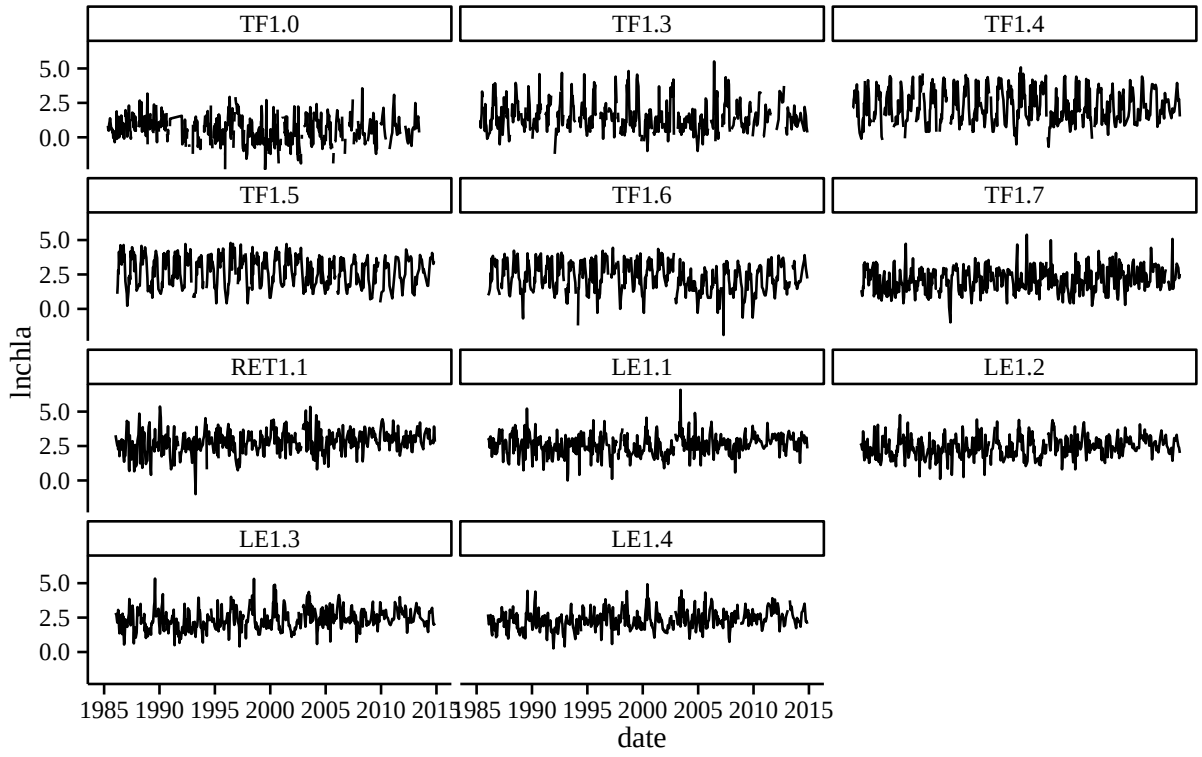
```
##
# merge chl and salinity data
# create fraction of freshwater

pax_data <- full_join(chl_tmp, sal_mrg, by = c('date', 'STATION')) %>%
  mutate(salff = 1 - sal/sal_ref_int)

ggplot(pax_data, aes(x = date, y = salff, group = STATION)) +
  geom_line() +
  theme_classic() +
  geom_hline(yintercept = 0, colour = 'blue', linetype = 'dashed') +
  facet_wrap(~ STATION, ncol = 3)

ggplot(pax_data, aes(x = date, y = lnchla, group = STATION)) +
  geom_line() +
  theme_classic() +
  facet_wrap(~ STATION, ncol = 3)
```


TF1.0
TF1.3
TF1.4
TF1.5
TF1.6
TF1.7
RET1.1
LE1.1
LE1.2
LE1.3
LE1.4
salff
1.0
0.5
0.0
−0.5
1985 1990 1995 2000 2005 2010 2015
date


TF1.0
TF1.3
TF1.4
TF1.5
TF1.6
TF1.7
RET1.1
LE1.1
LE1.2
LE1.3
LE1.4
lnchla
5.0
2.5
0.0
1985 1990 1995 2000 2005 2010 2015
date