**Code book for SET data processing**

1. Separate pin readings from all summaries and extraneous stuff. With “SETdataspreadsheet\_With\_NERR\_Stations\_Baseline\_MASTER.xls:
   1. Copy the first worksheet, “SET Data”, into a new file.
   2. Keep Columns A – last sample date.
   3. Keep rows starting at 103, which contains the date headings. Delete everything else.
   4. Currently there is one column that contains a pin identifier, and one column for each sample date. The pins are not fully labeled except the first one for a given site/arm position.
   5. Create and populate columns for:
      1. site (ex. “CLMAJ”)
         1. Juncus sites get split into upper and lower:
            1. UPJUN for JURO4-6
            2. LOJUN for JURO1-3
      2. site-platform (ex. “CLMAJ-1”)
      3. site-platform-position (ex. “CLMAJ-1-1L”)
      4. platform\_number (ex. “1”)
      5. arm\_position (ex. 1L, 5R, 7L, 3R)
      6. pin\_number (ex. 1-9)
      7. (some of these are redundant and this can probably be modified based on future data management practices, but it’s what I did initially)
   6. NEW 6/21/17: Delete 1st two dates of readings; new first date is 2/29/12 due to methods update. ONLY delete from the working analysis spreadsheet, NOT from the master file!
   7. Save as “SETs.csv”
2. Run “SET\_processing\_and\_summarizing.R” script on SETs.csv
   1. This will transform data into a long format, so date will be a variable instead of a column heading. The value will be the reading for each individual pin on each date.
   2. This script also generates summary data, grouped by site.platform (CLMAJ-1, CLMAJ-2, JURO-1, SPAL-3, etc) and date.
      1. mean (of the 36 pin readings for that site and date)
      2. stddev
      3. sdy-min (mean – stddev)
      4. sdy-max (mean + stddev)
      5. stderr
      6. seymin
      7. seymax
   3. This script then saves three data frames into the file “sets.grouped.R”:
      1. sets (the long file generated from the csv)
      2. sets\_platform\_date (same as sets, only grouped by site.platform and date)
      3. pin\_summary (data frame with mean, sd, etc.)
   4. Lower in the script, this also generates plots:
      1. scatter plots with a point for mean pin reading and error bars (+/- one standard *deviation*) for each platform by date.
      2. scatter plots with a point for mean pin reading and error bars (+/- one standard *error*) for each platform by date.
      3. boxplots showing the spread of pin readings at each platform by date.
3. “SET\_change\_calculations\_and\_plotting”:
   1. Imports “sets.grouped.R” from previous script.
   2. Uses the split-apply-combine strategy to:
      1. Split the pin summary data frame by platform
      2. Subtract the first (mean) reading for each platform from all subsequent (mean) readings and call this “change”
      3. Combine back into one big data frame
      4. Make plots of change over time by platform
      5. Generate a line representing sea level rise (local rate 3mm/yr) and put that on the plots for a visual comparison
   3. Generates linear models for change over time at each site.platform
      1. “modelcoef” is a data frame that gives intercept, slope, and p-values (among other things). It does not show confidence intervals by default, but in this script, it is set to give the 95% CI. Test-output is in the file “SETregressions2017-06-13.csv”.
         1. site.platform
         2. term (intercept or date)
         3. estimate
         4. std.error
         5. statistic
         6. p.value
         7. Calculated columns:
            1. mm.yr: Because of how R handles dates, slope comes out in mm/day. In this column, that has been converted to mm/year for easier interpretation. This accounts for leap years.
            2. conf.low.yr: lower end of 95% CI, converted to mm/yr
            3. conf.high.yr: higher end of 95% CI, converted to mm/yr
         8. see <https://rdrr.io/cran/broom/man/lm_tidiers.html> for more on the broom package and its output.
      2. “modelsummary” is a data frame that gives R^2, adjusted R^2, and p-values (among other things). Test-output is in the file “SETregressions2\_2017-06-13.csv”.
         1. site.platform
         2. r.squared
         3. adj.r.squared
         4. sigma
            1. residual standard error (summary(test.model)$sigma produces the same value labeled as “residual standard error” when printing summary(test.model) )
            2. (“the square root of the estimated variance of the random error”, from ?summary.lm)
            3. sqrt(sum(output$residuals^2)/output$df.residuals) from <https://stackoverflow.com/questions/15254543/obtain-residual-standard-errors-of-an-mlm-object-returned-by-lm>
         5. statistic
         6. p.value
         7. df
         8. logLik
         9. AIC
         10. BIC
         11. deviance: this is the residual sum of squares, I THINK. It matches the sum of squaring all the residuals in the lm output on an individual lm run.
         12. df.residual
      3. see <https://rdrr.io/cran/broom/man/lm_tidiers.html> for more on the broom package and its output.
      4. For reference on generating linear models by groups using dplyr and broom, see these answers on stackoverflow:
         1. <https://stackoverflow.com/questions/22713325/fitting-several-regression-models-with-dplyr>
         2. <https://stackoverflow.com/questions/1169539/linear-regression-and-group-by-in-r>
      5. Sure would be nice to add these to plots in an automated way. Script shows an individual site with SLR line and regression line overlaid.