loadcell

`loadcell` package provides tools for cleaning and parsing haul and endline data from load cell vertical line data collected for the Maine Department of Marine Resources.

Installing loadcell

`loadcell` is a local package, and requires `devtools` for installation.

First, install `devtools` with the following command:

```
install.packages('devtools')
```

Then, install the load cell package with the following command, substituting the local path to the `loadcell` package directory. This will install the package along with the several dependencies.

```
devtools::install_local(path='X:\\R\\Internal_Packages\\loadcell', dependencies = T)
```

Basic function use

Loading CSVs

A sample CSV of actual load cell data has been included in the package directory and will be used in this example. To load load cell CSV files from a directory, the `load_csvs` function can be used. The function will scan the current working directory, or optionally a directory can be passed with the `dir` argument. Directories are scanned recursively (CSVs in subdirectories will be included). In this example, my working directory is set to `example_data` in the package directory, so the single example CSV `17348465-2018.11.15-09.30.24-20.csv` will be loaded. The CSV is loaded to an object `csv_data` - using the `str` function will show the structure of this object. For more details, see the function documentation.

```
  # Set wd to example_data folder
setwd("X:\\Bio Monitor\\LoadCells\\R\\loadcell\\example_data")
# Load loadcell
library(loadcell)
# Load CSV data
csv_data <- load_csvs()
# Examine
str(csv_data)
#> List of 1
#>  $ 17348465-2018.11.15-09.30.24-20.csv:List of 8
#>   ..$ data    :'data.frame': 26556 obs. of  4 variables:
#>   .. ..$ TimeStamp: chr [1:26556] "2018-11-15-09:30:25.349" "2018-11-15-
09:30:25.704" "2018-11-15-09:30:26.038" "2018-11-15-09:30:26.392" ...
#>   .. ..$ Seconds  : num [1:26556] 0.375 0.73 1.063 1.417 1.761 ...
#>   .. ..$ Load     : int [1:26556] 28 28 28 28 28 29 29 29 29 29 ...
#>   .. ..$ Raw      : num [1:26556] 0.0174 0.0174 0.0174 0.0174 0.0175 ...
#>   ..$ sn      : chr "17348465"
#>   ..$ traps   : num 20
#>   ..$ start_dt: POSIXlt[1:1], format: "2018-11-15 09:30:25"
#>   ..$ end_dt  : POSIXlt[1:1], format: "2018-11-15 14:54:37"
#>   ..$ seconds : num 19452
#>   ..$ max_load: int 4205
#>   ..$ min_load: int 0
#>  - attr(*, "class")= chr "LoadCellData"
```

Parsing Hauls

After CSVs are loaded, hauls can be parsed. The `parse_hauls` function handles this and has several options for how hauls should be delimited from load cell data. More details on these parameters is in the function help. In this example, `split = T` is being used to indicate the CSVs need to be split into separate hauls. If a fisherman created a separate CSV file for each haul, `split = F` would be used to simply number the hauls.

```
  # Parse hauls from LoadCellData class object
haul_data <- parse_hauls(csv_data, split = T, # Split hauls within CSV
                         min_load = 40, # Remove data < 40LBF
                         min_time = 30, # Minimum 30 seconds haul length
                         min_gap = 30, # Minimum 30 seconds between hauls
                         pass = 30) # If a CSV is less than 30 seconds long,
                                    # assume it is a single haul and do not split
# Examine the first haul structure
str(haul_data[[1]])
#> List of 9
#>  $ haul    : num 1
#>  $ data    :'data.frame':    11 obs. of  5 variables:
#>   ..$ TimeStamp: chr [1:11] "2018-11-15-09:41:32.064" "2018-11-15-
09:41:32.756" "2018-11-15-09:41:37.247" "2018-11-15-09:41:51.869" ...
#>   ..$ Seconds  : num [1:11] 667 668 672 687 687 ...
#>   ..$ Load     : int [1:11] 44 40 41 40 40 43 45 52 64 68 ...
#>   ..$ Raw      : num [1:11] 0.0221 0.021 0.0212 0.0209 0.0209 ...
#>   ..$ Index    : int [1:11] 1 2 3 4 5 6 7 8 9 10 ...
#>  $ sn      : chr "17348465"
#>  $ traps   : num 20
#>  $ start_dt: POSIXlt[1:1], format: "2018-11-15 09:41:32"
#>  $ end_dt  : POSIXlt[1:1], format: "2018-11-15 09:42:11"
#>  $ seconds : num 39
#>  $ max_load: int 68
#>  $ min_load: int 40
```

Looking at the structure of `haul_data`, note that the single CSV has been split into 15 separate hauls based on the defined arguments.

Peak Analysis

Peaks analysis of haul data can be done with the `parse_peaks` function. This function will attempt to conduct peak analysis on each haul in an object of class `LoadCellHauls` created by the `parse_hauls` function. For more details on the arguments that can be input into this function, see the function help.

The goal of peak analysis is to identify the first major peak in the data that would typically occur when the first trap in a trawl reached the hauler. This first peak represents that maximum load on the vertical line that occured for the haul.

```
  # Parse peaks from LoadCellHauls class object
peaks <- parse_peaks(data = haul_data,
                     span=.05, # Percent smoothing to apply
                     peakdist=10, # Minimum 10 samples between peaks
                     peakheight=200) # Minimum peak height of 200 LBF
#> Loess smoothing was unsuccessful for haul 17348465-2018-11-15-1. Skipping to
next haul...
#> Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
#> parametric, : k-d tree limited by memory. ncmax= 200
#> Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
#> parametric, : k-d tree limited by memory. ncmax= 281
```

```
#> Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
#> parametric, : k-d tree limited by memory. ncmax= 246
#> Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
#> parametric, : k-d tree limited by memory. ncmax= 200
#> No peaks found for haul ID 17348465-2018-11-15-15
# Examine third haul peak analysis
str(peaks[[3]])
#> List of 17
#>  $ haul            : num 3
#>  $ data            :'data.frame':    164 obs. of  5 variables:
#>   ..$ TimeStamp: chr [1:164] "2018-11-15-09:50:40.059" "2018-11-15-
09:50:40.416" "2018-11-15-09:50:41.094" "2018-11-15-09:50:41.440" ...
#>   ..$ Seconds  : num [1:164] 1215 1215 1216 1216 1217 ...
#>   ..$ Load     : int [1:164] 47 40 47 61 66 66 66 67 75 80 ...
#>   ..$ Raw      : num [1:164] 0.0232 0.021 0.0231 0.0273 0.0288 ...
#>   ..$ Index    : int [1:164] 1 2 3 4 5 6 7 8 9 10 ...
#>  $ sn              : chr "17348465"
#>  $ traps           : num 20
#>  $ start_dt        : POSIXlt[1:1], format: "2018-11-15 09:50:40"
#>  $ end_dt          : POSIXlt[1:1], format: "2018-11-15 09:51:37"
#>  $ seconds         : num 57
#>  $ max_load        : int 1385
#>  $ min_load        : int 40
#>  $ span            : num 0.05
#>  $ peakdist        : num 10
#>  $ peakheight      : num 200
#>  $ peak_analysis   : logi TRUE
#>  $ smoothed_peaks  :'data.frame':    10 obs. of  2 variables:
#>   ..$ Index: num [1:10] 105 85 159 32 132 68 115 148 15 47
#>   ..$ Load : num [1:10] 1322 1313 851 844 683 ...
#>  $ smoothed_valleys:'data.frame':    10 obs. of  2 variables:
#>   ..$ Index: num [1:10] 7 60 144 46 19 118 154 93 72 82
#>   ..$ Load : num [1:10] 66 108 163 238 268 ...
#>  $ smoothed        :'data.frame':    164 obs. of  2 variables:
#>   ..$ Index: int [1:164] 1 2 3 4 5 6 7 8 9 10 ...
#>   ..$ Load : num [1:164] 43.4 46 50.9 58.1 64.9 ...
#>  $ actual_peaks    :'data.frame':    10 obs. of  2 variables:
#>   ..$ Load : int [1:10] 1385 1371 1385 852 685 692 1385 528 392 296
#>   ..$ Index: int [1:10] 105 85 105 32 132 68 105 147 15 48
```

Plotting data

Peaks can now be plotted with the `plot_hauls` function. Plots can be output to either the console or a PDF file. In this example, the hauls will be output to a PDF in the package folder called `TEST_PLOTS.pdf`

```
  # Set wd to example_output folder
setwd("X:\\Bio Monitor\\LoadCells\\R\\loadcell\\example_output")
# Build plots for hauls
plots <- plot_hauls(data = peaks, fileout = "TEST_PLOTS")
```

Exporting parsed data to CSVs

The `export_loadcell` function will export an object of class `LoadCellHauls` or `LoadCellPeaks` to a series of CSVs. A more in depth description of output is included in the function help. In this example, the data has been exported to a series of CSVs in the `example_output` folder in the package directory.

```
  # Set wd to example_output folder
setwd("X:\\Bio Monitor\\LoadCells\\R\\loadcell\\example_output")
# Export
export_loadcell(peaks, prefix = "TEST")
#> [1] TRUE
```