— title: "Flooding Events" author: "Andrew Brown" date: "November 23, 2015" output: pdf$_document$−−

```
r Read in matric potential, temperature and event data foo=read.csv("C:/Campbellsci/PC200W/CR1000-AB2_Table1.csv")Tensiometerandtemperaturetimeseriesevents = read.csv("ct_alfalfa_events.csv")Floodeventtimesandread.csv("tensio_events.csv")Mappingoftensiometer IDtoplot, treatmentand"active"eventstensio_calib = read.csv("ten
===SETUP=== PT_id = 2 : 21Rangeforcolumnscontainingpressuretransducerdata(20transducers)T_id = 43 : 51Rangeforcolumnscontainingtemperaturedata(9thermistors)major_x = "1week"Incrementbetweenlabeledgridli"1day"Incrementbetweenminorgridlinesorigin_time = "01/26/201510 : 00"Originfortimeaxis
Formatting TIMESTAMP = as.POSIXct(strptime(foo[,1],"event_start = as.POSIXct(strptime(events[, 4], "event_en as.POSIXct(strptime(events[, 5], "time_test = ts(TIMESTAMP, frequency = length(TIMESTAMP))Createatimes as.POSIXct(strptime(origin_time, "index = seq(1, length(TIMESTAMP))tindex = index∗15Eachrecordis15minutes as.numeric(substr(TIMESTAMP, 12, 13))Parseouthourofday
time_map = cbind(index, tindex, TIMESTAMP)TODOaddday, hour, eventtimestamps?
Data transformations keep=((as.numeric(TIMESTAMP)-as.numeric(origin) ¿ 0)) TIMESTAMPa=TIMESTAMP[kee TIMESTAMPn=seq(0,length(TIMESTAMPa))*15
foo2=foo COPY loaded data foo2[,PT_id+1] = (foo[, PT_id+1]−370)/1.742convertmVtombar(cmH2O)data = foo2[keep, ]Trimthedatatostartatthetimeaxisorigin(definedabove)
matric_head = data[, PT_id+1]makesomemoreconvenientframesandcolumnnamesnames(matric_head) = seq(1, 20)matric_std = data[, PT_id + 21]
temp_sensor = data[, T_id + 30]makesomemoreconvenientframesandcolumnnames
```

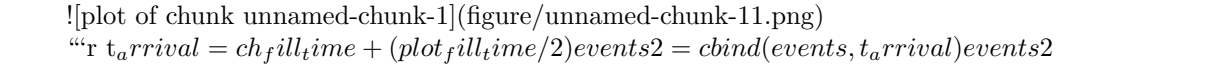
```
Error: undefined columns selected
r names(temp_sensor) = seq(1, 20)
Error: object 'temp_sensor' not found
r temp_std = data[, T_id + 21]
Error: undefined columns selected
r subsetTimes=function(time_start, time_end, buf = 3600)Returnsdatarecordscorrespondingtospecifiedtimeinterv
```

makePOSIXTime=function(tee) return(as.POSIXct(strptime(tee,"

Calculate surface boundary conditions

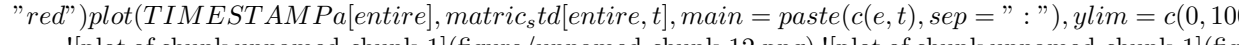
Assumed geometric "constants" for all 5 plots - can be treated on an individual basis also ch_depth = 30mean(c(20, 21, 25, 21, 25, 23, 22, 22, 20, 25, 22, 22, 22, 22, 19, 22, 23, 21, 24, 22, 22, 21, 20, 21, 22, 20, 22))22cmverticalch_width 75mean(c(2.5, 2.5, 2.6, 2.6, 2.5, 2.5)) ∗ 30.48plot_width = 600mean(c(20, 18, 19, 20, 22)) ∗ 30.48plot_length = 150050∗30.48gal_per_cc = 0.000264172ch_stor_gal = ((2∗ch_width)∗plot_length∗(ch_depth/2))∗gal_per_cgallonsofstorageinboth plot_length/2∗30.48tensiometersat1/2lengthofplottensio_clust_w = plot_width/2∗30.48plot_stor_gal = ((plot_length∗ plot_width) + (plot_length ∗ ch_width ∗ 2)) ∗ (ch_depth/2) ∗ gal_per_ctotal_stor_gal = ch_stor_gal + plot_stor_gal

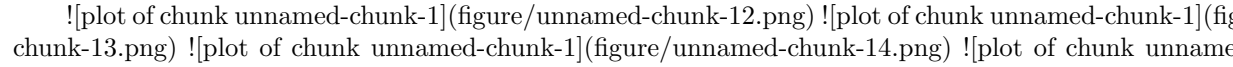
```
ch_fill_time = 0plot_fill_time = 0efficiency = 0for(ein1 : length(eventsDate)) flowrate=eventsFlow.rate[e]ch_fill_timch_stor_gal/flowrateplot_fill_time[e] = plot_stor_gal/flowrateefficiency[e] = plot_fill_time[e]/eventsminutes[e]
plot(total_stor_gal ∗ (2 − efficiency), eventsgallons)
```




```
r t_arrival = ch_fill_time + (plot_fill_time/2)events2 = cbind(events, t_arrival)events2
```

for(p in 1:5) loop through 5 monitored plots plot=tensio[p,] tid=c(plots1, plots2,plotd1, plotd2) eid=seq(ploteventsta par(mfcol=c(length(tid),length(eid))) for(e in eid) start_time = events[e,]START get event start time (water on) off_time = events[e,]END get water off time end_time = events[e+1,]START get start of next event (AKA end of current) entire=subsetTimes(start_time, end_time)minutes = seq(0, length(entire) − 1) ∗ 15irrig = subsetTimes(start_time, off_time, buf = 0)irr_start = which(entire == irrig[1])irr_stop = which(entire == irrig[length(irrig)])for(tintid)plot(minutes, matric_head[entire, t], main = paste(c(e, t), sep = " : "), ylim = c(0, 300))ab "red")plot(TIMESTAMPa[entire], matric_std[entire, t], main = paste(c(e, t), sep = " : "), ylim = c(0, 100))






```
Error: figure margins too large
```