Prophet Customer Forecast

## Introduction

This notebook will outline the steps to produce the prophet forecast, incorporating historical as well as future data for catalog circulation, marketing investment, and the J.Jill promotional calendar. The first step will be loading the required libraries to handle our data in R.

## Loading the Data

There are 4 required files used to build this forecast. They are:

1. Historical customer counts, monthly
2. Historical and future investment numbers, monthly
3. Historical and future catalog circulation, monthly
4. Historical and future promotional calendar

First we will investigate the customer data set.

### Customer Data

# sourced from alteryx workflow: Create Customer Data Set on latest masterXXk.yxdb file  
  
proph = read.csv('processed\_customers.csv')

### Investment Data

# sourced from Maria David, manually processed (no alteryx workflow)  
  
inv = read.csv('processed\_investments.csv')

### Catalog Circulation Data

# sourced from alteryx workflow: Prep Catalog History 20180613 on Jim's latest circulation calendar  
circ = read.csv('processed\_circ.csv')

### Promotional Calenendar Data

# sourced from alteryx workflow: Prop Promo File 20180717 on Kaitlyn's latest promo calendar  
promos = read.csv('processed\_promos.csv')

Now that we have all of the data prepped and ready for modeling, we can merge it into one larger file.

### Merge training data

regress = merge(proph, inv, by=c('FISCAL\_YR', 'FISCAL\_MO', 'Segment\_Channel'), all.y=TRUE)  
regress$ds <- as.yearmon(paste(regress$FISCAL\_YR, regress$FISCAL\_MO, sep = "-"))  
setnames(regress, 'monthly\_customers', 'y')  
  
regress2 = merge(regress, promos, by = c('FISCAL\_YR', 'FISCAL\_MO', 'FISCAL\_QTR', 'Segment\_Channel'), all.y=TRUE)  
regress3 = merge(regress2, circ, by = c('FISCAL\_YR', 'FISCAL\_MO', 'FISCAL\_QTR', 'Segment'), all.y=TRUE)  
regress3 = as.data.table(regress3)  
  
train = regress3[FISCAL\_YR >= 2016]

### Define the forecasting Function

The primary function used to build this forecast is called make\_forecast defined below. It takes two arguments, a data frame (df) as well as a threshold (threshold). The threshold is expected to be a year for which we want to forecast (i.e. 2019 or 2020). If you want to forecast for 2020, you should set the threshold to 2020, otherwise use 2019.

The forecast works by instantiating a prophet object (m) and setting a variety of parmeters relating to holidays, seasonality, and changepoints. Once the prophet object has been instantiated, a number of regressor variables can be added (any covariate column in the data frame argument can be incorporated as a regressor to the model.)

make\_forecast <- function(df, threshold) {  
 #threshold = 2019  
 #df = regress2[Segment\_Channel == 'React\_Direct1']  
 df = as.data.table(df)  
 df = df[FISCAL\_YR <= threshold]  
 m <- prophet(seasonality.mode = 'additive', holidays = holidays, holidays.prior.scale = .05, seasonality.prior.scale = 5, changepoint.prior.scale = .034, weekly.seasonality = FALSE, daily.seasonality = FALSE, yearly.seasonality = FALSE)  
 m <- add\_regressor(m, name = 'circ\_score')  
 #m <- add\_regressor(m, name = 'investment')  
 #m <- add\_regressor(m, name = 'peel\_off\_depth')  
 ##m <- add\_regressor(m, name = 'post\_card\_depth')  
 #m <- add\_regressor(m, name = 'free\_ship\_depth')  
 #m <- add\_regressor(m, name = 'flash\_sale\_depth')  
 #m <- add\_regressor(m, name = 'global\_depth')  
 #m <- add\_regressor(m, name = 'fp\_entire\_depth')  
 #m <- add\_regressor(m, name = 'sp\_entire\_depth')  
 m <- add\_seasonality(m, name='yearly', period=365.25, fourier.order=10, prior.scale = 0.05)  
 #m <- add\_seasonality(m, name = 'quarterly', period = 365.25/4, fourier.order = 5, prior.scale = 15)  
 #m <- add\_seasonality(m, name = 'monthly', period = 365.25/12, fourier.order = 3, prior.scale = 10)  
 m <- fit.prophet(m, df[FISCAL\_YR <= (threshold - 1)])  
 future <- make\_future\_dataframe(m, periods = 12, freq = 'month')  
 future$circ\_score = df$circ\_score  
 #future$investment = df$investment # FIXME this needs to be correctly specified  
 #future$peel\_off\_depth = df$peel\_off\_depth  
 ##future$post\_card\_depth = df$post\_card\_depth  
 #future$free\_ship\_depth = df$free\_ship\_depth  
 #future$flash\_sale\_depth = df$flash\_sale\_depth  
 #future$global\_depth = df$global\_depth  
 #future$fp\_entire\_depth = df$fp\_entire\_depth  
 #future$sp\_entire\_depth = df$sp\_entire\_depth  
 fcst <- predict(m, future)  
 prophet\_plot\_components(m, fcst)  
 return(fcst)  
}

Note that the function takes in a vector called holidays. We will need to define this vector to contain information on all of the J.Jill holidays. We do this manually by referencing the J.Jill fiscal calendar.

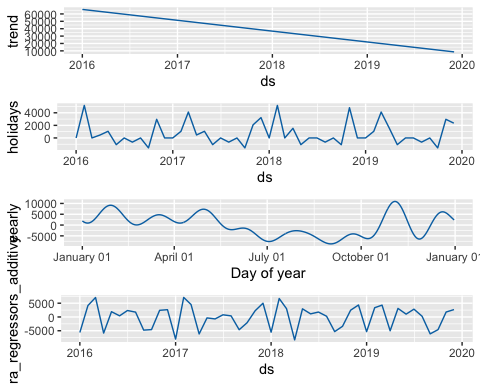
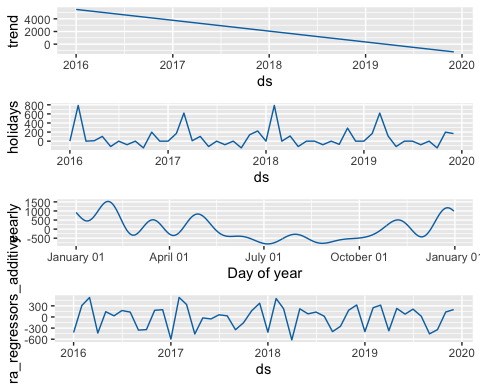
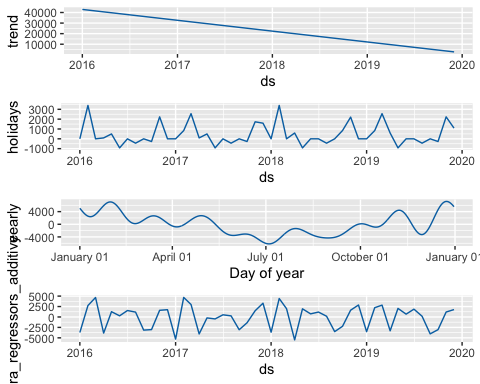
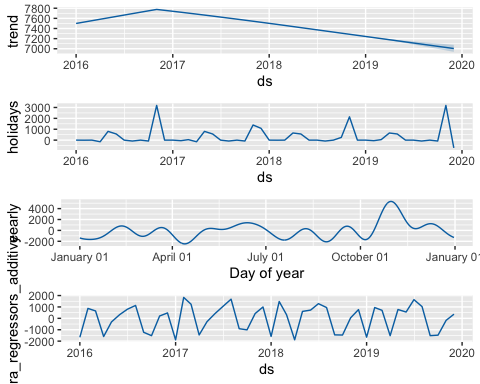
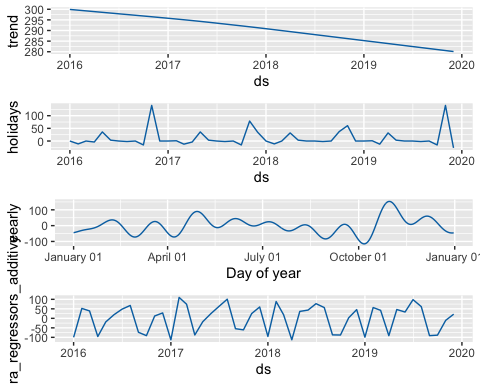
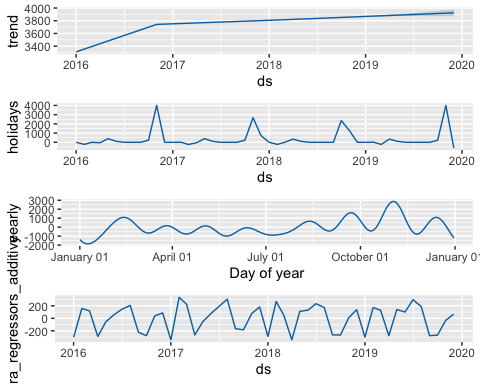
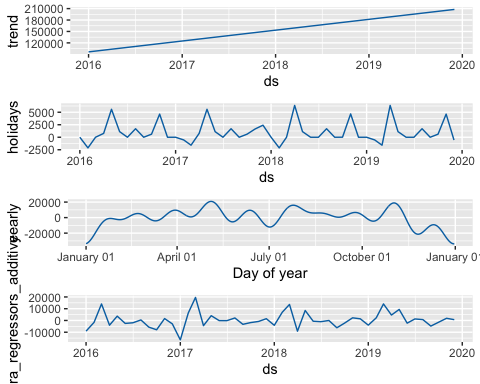
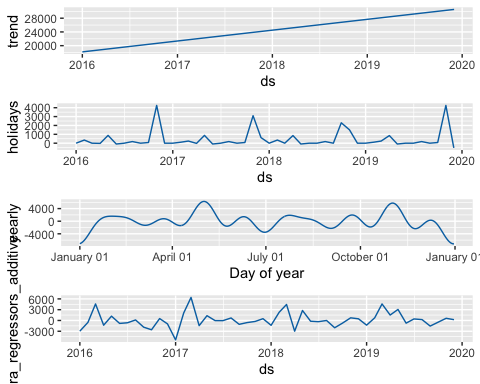
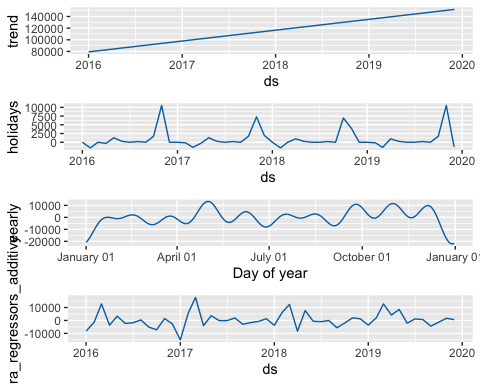
# HOLIDAYS  
new\_years <- tibble(  
 holiday = 'new\_years',  
 ds = as.Date(c('2017-12-01', '2018-11-01',  
 '2019-12-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
womens\_day <- tibble(  
 holiday = 'womens\_day',  
 ds = as.Date(c('2016-02-01', '2017-02-01', '2018-02-01',  
 '2019-02-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
good\_friday <- tibble(  
 holiday = 'good\_friday',  
 ds = as.Date(c('2016-02-01', '2017-03-01', '2018-02-01',  
 '2019-03-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
easter <- tibble(  
 holiday = 'easter',  
 ds = as.Date(c('2016-02-01', '2017-03-01', '2018-02-01',  
 '2019-03-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
mothers\_day <- tibble(  
 holiday = 'mothers\_day',  
 ds = as.Date(c('2016-04-01', '2017-04-01', '2018-04-01',  
 '2019-04-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
memorial <- tibble(  
 # on average +17k customers  
 holiday = 'memorial',  
 ds = as.Date(c('2016-05-01', '2017-05-01', '2018-04-01',  
 '2019-04-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
independence <- tibble(  
 # on average +5k customers  
 holiday = 'independence',  
 ds = as.Date(c('2016-06-01', '2017-06-01', '2018-05-01',  
 '2019-05-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
labor <- tibble(  
 # on average +20k customers  
 holiday = 'labor',  
 ds = as.Date(c('2016-08-01', '2017-08-01', '2018-08-01',  
 '2019-08-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
thanksgiv <- tibble(  
 holiday = 'thanksgiv',  
 ds = as.Date(c('2016-10-01', '2017-10-01', '2018-10-01',  
 '2019-10-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
black\_fri <- tibble(  
 holiday = 'black\_fri',  
 ds = as.Date(c('2016-10-01', '2017-10-01', '2018-10-01',  
 '2019-10-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
cyber\_monday <- tibble(  
 holiday = 'cyber\_monday',  
 ds = as.Date(c('2016-11-01', '2017-11-01', '2018-10-01',  
 '2019-11-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
christ\_eve <- tibble(  
 holiday = 'christ\_eve',  
 ds = as.Date(c('2016-11-01', '2017-11-01', '2018-11-01',  
 '2019-11-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
christ <- tibble(  
 holiday = 'christ',  
 ds = as.Date(c('2016-11-01', '2017-11-01', '2018-11-01',  
 '2019-11-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
nye <- tibble(  
 holiday = 'nye',  
 ds = as.Date(c('2016-11-01', '2017-12-01', '2018-11-01',  
 '2019-11-01')),  
 lower\_window = 0,  
 upper\_window = 1  
)  
  
  
  
  
  
  
holidays <- bind\_rows(nye, christ, cyber\_monday, christ\_eve,black\_fri, thanksgiv, labor, independence, memorial, mothers\_day, easter, good\_friday, womens\_day, new\_years)

Now that our function is defined and our holidays are coded, the next step is to produce a forecast for each of the segment channels in question. We will use R’s built in dplyr package to do this programmatically.

### Call our function for each segment channel in our dataframe

This block of code will run our make forecast function for 2019 9 times (note that we are grouping by Segment Channel). This means the forecast will run once for each segment channel that exists in our data frame.

fcst = train %>%   
 group\_by(Segment\_Channel) %>%  
 do(make\_forecast(., 2019)) %>%   
 dplyr::select(ds, Segment\_Channel, yhat)



Now we have a forecast at the monthly level. The next step will be to calculate how this monthly forecast will translate to quarterly and yearly totals. We will define a function calc\_year\_end to do this for us. This function will use two files, monthly\_pacing taken from percentages.txt and quarter\_totals taken from quarter\_totals.txt. Both of these files were sourced from alteryx workflow: percentages.yxmd

monthly\_pacing = fread("percentages.txt", sep= '|')  
setnames(monthly\_pacing, 'Fiscal\_Mo', 'FISCAL\_MO')  
  
  
  
quarterly\_pacing = fread("quarter\_totals.txt", sep= '|')  
quarterly\_pacing[, quarter := ifelse(FISCAL\_MO <= 3, 1, ifelse(FISCAL\_MO <= 6, 2, ifelse(FISCAL\_MO <= 9, 3, 4)))]  
quarterly\_pacing[, quarter\_sum := sum(cust\_month), by=c("quarter", "Customer\_Type")]  
quarterly\_pacing[, table := cust\_quarter/quarter\_sum]  
quarter\_table = unique(quarterly\_pacing[, c("Customer\_Type", "quarter", "table", "quarter\_sum")])  
setnames(quarter\_table, "Customer\_Type", "Segment\_Channel")  
  
  
calc\_year\_end <- function(cast\_dat, year, hist) {  
 et = as.data.table(cast\_dat)  
 et[, FISCAL\_MO := month(ds)]  
 et[, quarter := ifelse(FISCAL\_MO %in% c(1,2,3), 1, ifelse(FISCAL\_MO %in% c(4,5,6), 2, ifelse(FISCAL\_MO %in% c(7,8,9), 3,4)))]  
 et[, day := format(as.Date(ds,format="%Y-%m-%d"), "%d")]  
 et[, FISCAL\_YR := year(ds)]  
 et = et[ day == '01']  
 setnames(et, 'yhat', 'Used\_Forecast')  
   
   
   
 #wrap up  
 df2 = hist[, c('Segment\_Channel', 'FISCAL\_YR', 'FISCAL\_MO', 'y')]  
 et2 = merge(et, df2, by=c('Segment\_Channel', 'FISCAL\_MO', 'FISCAL\_YR'), all.x=TRUE)  
 et2[, Used\_Forecasta := ifelse(is.na(y), Used\_Forecast, y)]  
  
   
 year\_agg = merge(et2, monthly\_pacing, by = c("Segment\_Channel", "FISCAL\_MO"))  
 year\_agg = merge(year\_agg, quarter\_table, by = c("quarter", "Segment\_Channel"))  
 year\_agg[, Year\_Number2 := Used\_Forecast \* percent]  
 year\_agg[, Year\_Number2a := Used\_Forecasta \* percent]  
 year\_agg[, qnum := table\*Used\_Forecast]  
 year\_agg[, qnuma := table \* Used\_Forecasta]  
  
   
 return(year\_agg)  
}

### Examine results of forecasting run

final\_fcst = calc\_year\_end(fcst, 2019, regress3)   
final\_fcst = final\_fcst[FISCAL\_YR == 2019]

#estimates = c(1, 0.544586951, 0.366562835, 0.337894522)  
#q\_s = final\_fcst %>% group\_by(quarter, Segment\_Channel) %>% summarize(sum(qnuma))  
#q\_s  
#sum(q\_s\*estimates) #1921

Once we are happy with the output of our forecast, the final step is to write it to a csv file. The data from this csv file should be pasted into the Prophet Forecast Data tab of august\_2020\_v2 excel workbook. The data and forecast will update automatically after copying and pasting.

### Write out forecast to csv file

write.csv(final\_fcst, '~/Desktop/prophet\_aug.csv')