Wairakei Stochastic Simulation

Logan Wu

8/11/2018

# Preprocessing

Data is extracted and cleaned using Python. Some transformations such as datetimes to days since baseline are also done in Python, and saved into a config spreadsheet.

R reads the cleaned data from the spreadsheet and uses this to: \* Create a graph structure \* Make the data into a JAGS-readable format

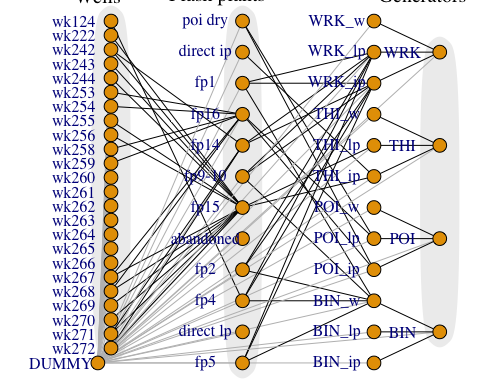
configsheets = excel\_sheets(configpath)  
for (sheet in configsheets) {  
 assign(sheet, read\_excel(configpath, sheet))  
}  
regression\_df = read\_excel(regdatapath) %>%  
 mutate(date\_numeric=ifelse(date\_numeric>0, date\_numeric, NA)) # remove dates before baseline

## Warning: package 'bindrcpp' was built under R version 3.4.4

well\_fp\_map = well\_fp\_map %>% drop\_na()  
  
today\_numeric = (Sys.time() - base\_datetime) %>% as.numeric()  
  
# assign unique facility IDs  
reg\_wells = unique(regression\_df$well)  
map\_wells = unique(well\_fp\_map$well)  
no\_data\_wells = map\_wells[!map\_wells %in% reg\_wells] # see which ones we're completely guessing for  
well\_names = c(reg\_wells) %>% unique() # c(reg\_wells, map\_wells) %>% unique()  
fp\_names = c(well\_fp\_map$fp, fp\_gen\_map$fp,fp\_constants$fp) %>% unique()  
fluid\_types = c('ip', 'lp', 'w')  
gen\_names = gen\_constants$gen %>% unique() %>% sort()  
ip\_gen\_names = paste(gen\_names, 'ip', sep='\_')  
lp\_gen\_names = paste(gen\_names, 'lp', sep='\_')  
w\_gen\_names = paste(gen\_names, 'w', sep='\_')  
dummy\_gen\_names = c(ip\_gen\_names, lp\_gen\_names, w\_gen\_names) %>% sort()  
all\_names = c('DUMMY', well\_names, fp\_names, dummy\_gen\_names, gen\_names)  
ids = 1:length(all\_names)  
names(ids) = all\_names  
  
# replace names in data with IDs  
regression\_df = regression\_df %>% mutate(well\_id=ids[well]) %>% select(-well)  
operating\_conditions = operating\_conditions %>% mutate(well\_id=ids[well]) %>% rename(whp\_pred=whp)  
fp\_constants = fp\_constants %>% mutate(fp\_id=ids[fp]) %>% select(-fp)  
gen\_constants = gen\_constants %>% mutate(gen\_id=ids[gen]) %>% select(-gen)  
well\_fp\_map = well\_fp\_map %>% mutate(well\_id=ids[well], fp\_id=ids[fp]) %>% select(-c(well, fp))  
fp\_gen\_map = fp\_gen\_map %>% mutate(fp\_id=ids[fp], gen\_ip\_id=ids[gen\_ip], gen\_lp\_id=ids[gen\_lp], gen\_w\_id=ids[gen\_w]) %>% select(-c(fp, gen\_ip, gen\_lp, gen\_w))

# Graph

# create connectivity matrix. i flows to j  
# wells to FPs  
v = matrix(0, nrow=length(ids), ncol=length(ids))  
v[1,-1] = 1  
for (i in 1:nrow(well\_fp\_map)) {  
 id\_i = well\_fp\_map[[i, 1]]  
 id\_j = well\_fp\_map[[i, 2]]  
 v[id\_i, id\_j] = 1  
}  
# send ip/lp/w flows to dummy gens  
for (i in 1:nrow(fp\_gen\_map)) {  
 id\_i = fp\_gen\_map[[i, 1]]  
 for (j in 2:ncol(fp\_gen\_map)) {  
 facility\_j = names(ids)[fp\_gen\_map[[i, j]]]  
 facility\_dummy\_j = paste(facility\_j, fluid\_types[j-1], sep='\_')  
 id\_j = ids[facility\_dummy\_j]  
 if (!is.na(id\_j)) {  
 v[id\_i, id\_j] = 1  
 }  
 }  
}  
# dummy gens to gens  
for (i in 1:nrow(gen\_constants)) {  
 id\_j = gen\_constants$gen\_id[i]  
 facility\_j = names(ids)[id\_j]  
 for (fluid in fluid\_types) {  
 facility\_dummy\_i = paste(facility\_j, fluid, sep='\_')  
 id\_i = ids[facility\_dummy\_i]  
 v[id\_i, id\_j] = 1  
 }  
}  
  
# convert form  
m = matrix(0, nrow=nrow(v), ncol=max(colSums(v)))  
rownames(m) = all\_names  
for (i in 1:nrow(v)) {  
 for (j in 1:ncol(v)) {  
 if (v[[i, j]]==1) {  
 m[j, sum(m[j,]>0)+1] = i  
 }  
 }  
}  
  
# generate coordinates  
dummy\_locs = data.frame(name='DUMMY', x=-0.1, y=0)  
well\_locs = data.frame(name=well\_names, x=0, y=seq(1, 1/(length(well\_names)-1), length.out=length(well\_names)))  
fp\_locs = data.frame(name=fp\_names, x=1, y=seq(0, 1, length.out=length(fp\_names)))  
gen\_dummy\_locs = data.frame(name=dummy\_gen\_names, x=2, y=seq(0, 1, length.out=length(dummy\_gen\_names)))  
gen\_locs = data.frame(name=gen\_names, x=2.5, y=seq(1/11, 10/11, length.out=length(gen\_names)))  
locs = rbind(dummy\_locs, well\_locs, fp\_locs, gen\_dummy\_locs, gen\_locs)  
locs$id = ids[locs$name]  
locs = locs %>% arrange(id)  
  
g = graph\_from\_adjacency\_matrix(v) %>%  
 set\_vertex\_attr('label', value=all\_names) %>%  
 set\_vertex\_attr('x', value=as.vector(locs$x)) %>%  
 set\_vertex\_attr('y', value=as.vector(locs$y)) %>%  
 set\_vertex\_attr('label.degree', value=pi) %>%  
 set\_vertex\_attr('size', value=8) %>%  
 as.undirected()  
E(g)$color = "black"  
E(g)[which(tail\_of(g, E(g))$label=="DUMMY")]$color = "grey"  
  
# png("../media/full\_network.png")  
par(mar=c(0,3,0,0), family="Times")  
plot(g, vertex.label.dist=3,  
 mark.groups = list(wells=ids[well\_names], fps=ids[fp\_names], gens=ids[gen\_names]),  
 mark.col = "#EEEEEE",  
 mark.border = NA)  
text(c(-1, -0.3, 0.4, 0.9), 1.15, c("Wells", "Flash plants", "Dummy gens", "Generators"), cex=1.25)



The dummy node is necessary because when indexing a subset of flows that go into a node, this subset cannot be empty. The dummy node has zero mass flowing out of it.

# Data

regression\_list = regression\_df %>% select(-c(date, h)) %>% as.list()  
operating\_conditions\_list = operating\_conditions %>% arrange(well\_id) %>% select(whp\_pred) %>% as.list()  
fp\_constants\_list = as.list(fp\_constants)  
gen\_constants\_list = as.list(gen\_constants %>% select(gen\_id, factor))  
facilities = data.frame(id=1:max(ids)) %>%  
 full\_join(operating\_conditions %>% rename(id=well\_id) %>% select(-well), by='id') %>%  
 full\_join(gen\_constants %>% select(factor, id=gen\_id), by='id') %>%  
 full\_join(fp\_constants %>% rename(id=fp\_id), by='id') %>%  
 mutate(mf\_pred=NA) %>%  
 mutate(n\_inflows=colSums(v))

## Warning: Column `id` has different attributes on LHS and RHS of join  
  
## Warning: Column `id` has different attributes on LHS and RHS of join  
  
## Warning: Column `id` has different attributes on LHS and RHS of join

facilities\_list = facilities %>% select(-id) %>% as.list()  
  
well\_ids = ids[well\_names]  
fp\_ids = ids[fp\_names]  
ip\_gen\_ids = ids[ip\_gen\_names]  
lp\_gen\_ids = ids[lp\_gen\_names]  
w\_gen\_ids = ids[w\_gen\_names]  
gen\_ids = ids[gen\_names]  
  
# insert production curve predictions  
prod = data.frame(whp\_prod=seq(7, 16, length.out=10),  
 well\_id\_prod=ids[production\_curve\_well])

## Warning in data.frame(whp\_prod = seq(7, 16, length.out = 10), well\_id\_prod  
## = ids[production\_curve\_well]): row names were found from a short variable  
## and have been discarded

prod\_list = prod %>% as.list()  
  
data = c(regression\_list, facilities\_list, prod\_list,  
 list(well\_ids=well\_ids, fp\_ids=fp\_ids, gen\_ids=gen\_ids,  
 ip\_gen\_ids=ip\_gen\_ids, lp\_gen\_ids=lp\_gen\_ids, w\_gen\_ids=w\_gen\_ids,  
 today\_numeric=today\_numeric, m=m, dummy=1))  
data$whp\_pred[is.na(data$whp\_pred)] <- mean(data$whp\_pred, na.rm=T)  
# data$date\_numeric\_c <- data$date\_numeric - today\_numeric

# Model

code = "  
data {  
 whp\_c <- whp - mean(whp)  
 whp\_c\_prod <- whp\_prod - mean(whp)  
 whp\_c\_pred <- whp\_pred - mean(whp)  
}  
model {  
 #######################################  
 # fit individual regressions to wells #  
 #######################################  
 for (i in 1:length(whp)) {  
 # elliptic  
 # mu2[i] <- beta\_whp[well\_id[i]] \* whp[i]^2 + beta\_date[well\_id[i]] \* date\_numeric[i]^2  
 # mu2[i] <- beta\_date[well\_id[i]]\*date\_numeric[i] \* sqrt(max(Intercept[well\_id[i]]-beta\_whp[well\_id[i]]\*whp[i]^2, 0)) + (beta\_date[well\_id[i]]\*date\_numeric[i])^2 + Intercept[well\_id[i]]-beta\_whp[well\_id[i]]\*whp[i]^2  
 # mu[i] <- sqrt(max(mu2[i], 0))  
 # exponential  
 # mu[i] <- beta\_whp[well\_id[i]] \* (Intercept[well\_id[i]] - whp[i]^2) ^ beta\_date[well\_id[i]]  
 # quadratic  
 # mu[i] <- Intercept[well\_id[i]] + beta\_whp[well\_id[i]] \* whp\_c[i] + beta\_whp2[well\_id[i]] \* whp\_c[i]^2 + beta\_date[well\_id[i]] \* date\_numeric[i] \* measurement\_error\_factor  
 # linear  
 mu[i] <- Intercept[well\_id[i]] + beta\_whp[well\_id[i]] \* whp\_c[i] + beta\_date[well\_id[i]] \* date\_numeric[i]  
  
 mf[i] ~ dnorm(mu[i], tau[well\_id[i]])  
 mf\_fit[i] ~ dnorm(mu[i], tau[well\_id[i]])  
 }  
 measurement\_error\_factor ~ dunif(0.9, 1.1)  
   
 # HIERARCHICAL  
 # fills in for any missing wells  
 for (j in well\_ids) {  
 Intercept[j] ~ dnorm(mu\_Intercept, tau\_Intercept)  
 beta\_whp[j] ~ dnorm(mu\_beta\_whp, tau\_beta\_whp)  
 # beta\_whp2[j] ~ dnorm(mu\_beta\_whp2, tau\_beta\_whp2)  
 beta\_date[j] ~ dnorm(mu\_beta\_date, tau\_beta\_date)  
 tau[j] ~ dgamma(1e-12, 1e-12)  
 sd[j] <- 1/sqrt(tau[j])  
 }  
 # fill in any missing dates  
 for (i in 1:length(whp)) {  
 date\_numeric[i] ~ dnorm(mu\_date\_numeric, tau\_date\_numeric)  
 }  
 mu\_date\_numeric ~ dnorm(0, 1e-12)  
 tau\_date\_numeric ~ dnorm(1e-12, 1e-12)  
   
 # set hyperparameters  
 mu\_Intercept ~ dnorm(0, 1e-12)  
 mu\_beta\_whp ~ dnorm(0, 1e-12)  
 # mu\_beta\_whp2 ~ dnorm(0, 1e-12)  
 mu\_beta\_date ~ dnorm(0, 1e-12)  
 tau\_Intercept ~ dgamma(1e-12, 1e-12)  
 tau\_beta\_whp ~ dgamma(1e-12, 1e-12)  
 # tau\_beta\_whp2 ~ dgamma(1e-12, 1e-12)  
 tau\_beta\_date ~ dgamma(1e-12, 1e-12)  
  
 #####################################  
 # production curve for verification #  
 #####################################  
 for (i in 1:length(whp\_prod)) {  
 # elliptic  
 # mf\_prod2[i] <- beta\_whp[well\_id\_prod[i]] \* whp\_c\_prod[i]^2 + beta\_date[well\_id\_prod[i]] \* today\_numeric^2  
 # mf\_prod[i] <- sqrt(max(mf\_prod2[i], 0)) + Intercept[well\_id\_prod[i]]  
  
 # mf\_prod2[i] <- beta\_date[well\_id\_prod[i]]\*today\_numeric \* sqrt(max(Intercept[well\_id\_prod[i]]-beta\_whp[well\_id\_prod[i]]\*whp\_c\_prod[i]^2, 0)) + (beta\_date[well\_id[i]]\*today\_numeric)^2 + Intercept[well\_id[i]]-beta\_whp[well\_id\_prod[i]]\*whp\_c\_prod[i]^2  
 # mf\_prod[i] <- sqrt(max(mf\_prod2[i], 0))  
  
 # exponential  
 # mf\_prod[i] <- beta\_whp[well\_id\_prod[i]] \* (Intercept[well\_id\_prod[i]] - whp\_c\_prod[i]^2) ^ beta\_date[well\_id\_prod[i]]  
 # quadratic  
 # mf\_prod[i] <- Intercept[well\_id\_prod[i]] + beta\_whp[well\_id\_prod[i]] \* whp\_c\_prod[i] + beta\_whp2[well\_id\_prod[i]] \* whp\_c\_prod[i]^2 + beta\_date[well\_id\_prod[i]] \* today\_numeric  
 # linear  
 mf\_prod[i] <- Intercept[well\_id\_prod[i]] + beta\_whp[well\_id\_prod[i]] \* whp\_c\_prod[i] + beta\_date[well\_id\_prod[i]] \* today\_numeric  
 }  
  
 ######################################################  
 # simple model to fill in missing enthalpy constants #  
 ######################################################  
 for (i in fp\_ids) {  
 # missing fp constants  
 hf\_ip[i] ~ dgamma(param[1], param[7])  
 hg\_ip[i] ~ dgamma(param[2], param[8])  
 hfg\_ip[i] ~ dgamma(param[3], param[9])  
 hf\_lp[i] ~ dgamma(param[4], param[10])  
 hg\_lp[i] ~ dgamma(param[5], param[11])  
 hfg\_lp[i] ~ dgamma(param[6], param[12])  
 }  
 for (i in c(1, well\_ids)) { h[i] ~ dgamma(param[13], param[14]) } # missing well constants  
 for (i in 1:14) { param[i] ~ dgamma(1e-12, 1e-12) } # uniform priors  
  
 ########################################  
 # make predictions (the stuff we want) #  
 ########################################  
 mf\_pred[dummy] <- 0 # dummy well  
 ip\_sf[dummy] <- 0  
 lp\_sf[dummy] <- 0  
 wf[dummy] <- 0  
   
 for (i in well\_ids) {  
 # elliptic  
 # mf\_pred2[i] <- beta\_whp[i] \* whp\_c\_pred[i]^2 + beta\_date[i] \* today\_numeric^2  
 # mf\_pred[i] <- sqrt(max(mf\_pred2[i], 0)) + Intercept[i]  
 # mf\_pred2[i] <- beta\_date[i]\*today\_numeric \* sqrt(max(Intercept[i]-beta\_whp[i]\*whp\_c\_pred[i]^2, 0)) + (beta\_date[i]\*today\_numeric)^2 + Intercept[i]-beta\_whp[i]\*whp\_c\_pred[i]^2  
 # mf\_pred[i] <- sqrt(max(mf\_pred2[i], 0))  
 # exponential  
 # mf\_pred[i] <- beta\_whp[i] \* (Intercept[i] - whp\_c\_pred[i]^2) ^ beta\_date[i]  
 # quadratic  
 # mf\_pred[i] <- Intercept[i] + beta\_whp[i] \* whp\_c\_pred[i] + beta\_whp2[i] \* whp\_c\_pred[i]^2 + beta\_date[i] \* today\_numeric  
 # linear  
 mf\_pred[i] <- Intercept[i] + beta\_whp[i] \* whp\_c\_pred[i] + beta\_date[i] \* today\_numeric  
 }  
 for (i in fp\_ids) {  
 mf\_pred[i] <- sum(mf\_pred[m[i,1:n\_inflows[i]]])  
 h[i] <- sum(mf\_pred[m[i, 1:n\_inflows[i]]] \* h[m[i, 1:n\_inflows[i]]]) / ifelse(mf\_pred[i]!=0, mf\_pred[i], 1)  
 ip\_sf[i] <- (h[i] - hf\_ip[i]) / hfg\_ip[i] \* mf\_pred[i]  
 lp\_sf[i] <- (hf\_ip[i] - hf\_lp[i]) / hfg\_lp[i] \* (mf\_pred[i] - ip\_sf[i])  
 total\_sf[i] <- ip\_sf[i] + lp\_sf[i]  
 wf[i] <- total\_sf[i]  
 }  
 # dummy gens and actual gens  
 for (i in ip\_gen\_ids) { mf\_pred[i] <- sum(ip\_sf[m[i, 1:n\_inflows[i]]]) }  
 for (i in lp\_gen\_ids) { mf\_pred[i] <- sum(lp\_sf[m[i, 1:n\_inflows[i]]]) }  
 for (i in w\_gen\_ids) { mf\_pred[i] <- sum(wf[m[i, 1:n\_inflows[i]]]) }  
 for (i in gen\_ids) {  
 mf\_pred[i] <- sum(mf\_pred[m[i,1:n\_inflows[i]]])  
 power[i] <- mf\_pred[i] / mu\_factor[i]  
 mu\_factor[i] ~ dunif(0.95\*factor[i], 1.05\*factor[i]) # uncertainty from email  
 }  
 total\_power <- sum(power[gen\_ids])  
}  
"  
  
vars = c(paste0('mf\_fit[', 1:length(data$whp), ']'),  
 paste0('mf\_pred[', 2:length(data$mf\_pred), ']'),  
 paste0('beta\_date[', data$well\_ids, ']'),  
 paste0('sd[', data$well\_ids, ']'),  
 paste0('power[', gen\_ids, ']'),  
 paste0('h[', fp\_ids, ']'),  
 paste0('mf\_prod[', 1:length(data$whp\_prod), ']'),  
 'total\_power',  
 paste0('mu\_', c('Intercept', 'beta\_whp', 'beta\_date')))  
n\_chains = 2  
burn\_in = 500  
n\_steps = 5000  
  
model = jags.model(textConnection(code), data, n.chains=n\_chains)

## Compiling data graph  
## Resolving undeclared variables  
## Allocating nodes  
## Initializing  
## Reading data back into data table  
## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 1565  
## Unobserved stochastic nodes: 902  
## Total graph size: 8015  
##   
## Initializing model

update(model, burn\_in)  
out = coda.samples(model, n.iter=round(n\_steps/n\_chains), variable.names=vars)  
outmatrix = as.matrix(out)  
outframe = as.data.frame(outmatrix) %>%  
 gather(key=facility, value=value) %>%  
 mutate(variable=gsub("\\[.\*$", "", facility), facility=parse\_number(facility, na=c("NA")))

## Warning: 20000 parsing failures.  
## row # A tibble: 5 x 4 col row col expected actual expected <int> <int> <chr> <chr> actual 1 4235001 NA a number mu\_Intercept row 2 4235002 NA a number mu\_Intercept col 3 4235003 NA a number mu\_Intercept expected 4 4235004 NA a number mu\_Intercept actual 5 4235005 NA a number mu\_Intercept  
## ... ................. ... ..................................... ........ ..................................... ...... ..................................... ... ..................................... ... ..................................... ........ ..................................... ...... .....................................  
## See problems(...) for more details.

outframe$facility = names(ids)[outframe$facility]

# Posteriors

g1 = ggplot(outframe %>% filter(facility %in% well\_names, variable=="mf\_pred", value>0), aes(x=value, fill=facility)) +  
 geom\_density(alpha=0.5, color=NA) + xlim(0, NA) +  
 labs(title="Posterior Well Mass Flows for 2018", x="Mass flow", y="Density", fill="Facility") +  
 ggsave('../media/mf\_wells.png', width=6, height=4, units='in')  
   
g2 = ggplot(outframe %>% filter(variable=="beta\_date"), aes(x=value, fill=facility)) +  
 geom\_density(alpha=0.5, color=NA) +labs(title="Posterior Decline Rate of Test Data", x="Beta\_Date", y="Density", fill="Facility") +  
 ggsave('../media/beta\_date.png', width=6, height=4, units='in')  
  
g3 = ggplot(outframe %>% filter(facility %in% fp\_names, variable=="mf\_pred", value>0), aes(x=value, fill=facility)) +  
 geom\_density(aes(y=..scaled..), alpha=0.5, color=NA) + xlim(0, NA) +   
 labs(title="Posterior Flash Plant Mass Flows for 2018", x="Mass flow")  
  
g4 = ggplot(outframe %>% filter(facility %in% gen\_names, variable=="mf\_pred", value>0), aes(x=value, fill=facility)) +  
 geom\_density(aes(y=..scaled..), alpha=0.5, color=NA) + xlim(0, NA) +   
 labs(title="Posterior Generator Mass Flows for 2018", x="Mass flow", y="Scaled density", fill="Facility") +  
 ggsave('../media/mf\_gens.png', width=6, height=4, units='in')  
  
g5 = ggplot(outframe %>% filter(facility %in% gen\_names, variable=="power", value>0), aes(x=value, fill=facility)) +  
 geom\_density(aes(y=..scaled..), alpha=0.5, color=NA) + xlim(0, NA) +   
 labs(title="Posterior Generator Power Output for 2018", x="Power", y="Scaled density", fill="Facility") +  
 ggsave('../media/power\_gens.png', width=6, height=4, units='in')  
  
tb6 <- outframe %>% filter(variable=="sd") %>% select(facility, value) %>%  
 mutate(well=factor(facility)) %>%  
 group\_by(well) %>%  
 summarise(Mean = mean(value),   
 `Lower 2.5%` = quantile(value, 0.025),   
 `Upper 97.5%` = quantile(value, 0.975)) %>%  
 mutate\_if(is.numeric, round, 3) %>%  
 inner\_join(regression\_df %>% mutate(well=factor(names(ids)[well\_id])) %>% group\_by(well) %>% summarise(n=n()))

## Joining, by = "well"

g6 = ggplot(outframe %>% filter(variable=="sd") %>% filter(facility %in% c('wk124', 'wk242', 'wk263', 'wk270', 'wk271')), aes(x=value, fill=facility)) +  
 geom\_density(alpha=0.5, color=NA) + coord\_cartesian(xlim=c(0, max(tb6$`Upper 97.5%`))) +  
 labs(title="Posterior Flow Deviation Estimates", x="Standard deviation", y="Density", fill="Facility") +  
 ggsave('../media/standard\_deviation.png', width=6, height=4, units='in')  
  
ggplotly(g1, tooltip=c('facility', 'value'))

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

ggplotly(g2, tooltip=c('facility', 'value'))

ggplotly(g3, tooltip=c('facility', 'value'))

ggplotly(g4, tooltip=c('facility', 'value'))

ggplotly(g5, tooltip=c('facility', 'value'))

ggplotly(g6, tooltip=c('facility', 'value'))

tb2 <- outframe %>% filter(variable=="beta\_date") %>% select(facility, value) %>%  
 mutate(well=factor(facility)) %>%  
 group\_by(well) %>%  
 summarise(Mean = mean(value),   
 `Lower 2.5%` = quantile(value, 0.025),   
 `Upper 97.5%` = quantile(value, 0.975)) %>%  
 mutate\_if(is.numeric, round, 3) %>%  
 inner\_join(regression\_df %>% mutate(well=factor(names(ids)[well\_id])) %>% group\_by(well) %>% summarise(n=n()))

## Joining, by = "well"

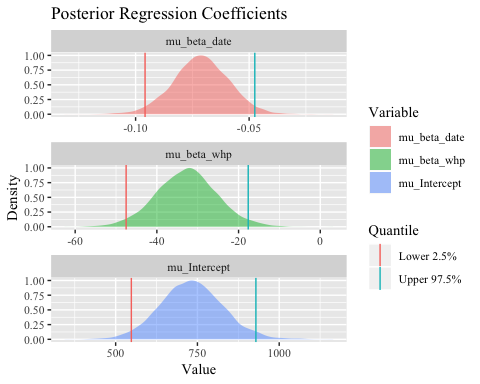
tb2[tb2$well %in% c('wk124', 'wk242', 'wk263', 'wk270', 'wk271'),] %>% knitr::kable()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| well | Mean | Lower 2.5% | Upper 97.5% | n |
| wk124 | -0.064 | -0.090 | -0.044 | 31 |
| wk242 | 0.014 | -0.017 | 0.044 | 28 |
| wk263 | -0.014 | -0.020 | -0.007 | 34 |
| wk270 | -0.067 | -0.143 | 0.009 | 5 |
| wk271 | -0.070 | -0.154 | 0.010 | 6 |

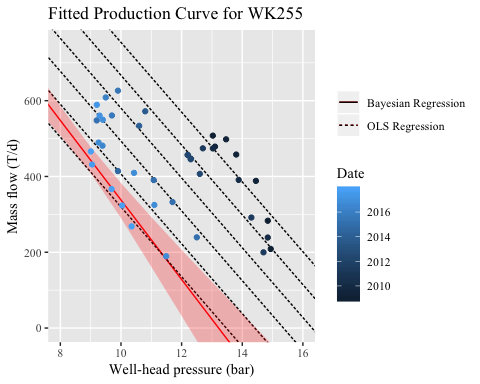
tb6[tb6$well %in% c('wk124', 'wk242', 'wk263', 'wk270', 'wk271'),] %>% knitr::kable()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| well | Mean | Lower 2.5% | Upper 97.5% | n |
| wk124 | 22.788 | 17.746 | 29.879 | 31 |
| wk242 | 147.618 | 111.449 | 198.527 | 28 |
| wk263 | 14.049 | 11.006 | 18.085 | 34 |
| wk270 | 35.930 | 15.060 | 92.126 | 5 |
| wk271 | 105.485 | 48.771 | 238.000 | 6 |
| # Hyper- | parameters |  |  |  |

hp.df <- outframe %>% filter(str\_detect(variable, 'mu\_'))  
hp.quantiles <- hp.df %>%  
 group\_by(variable) %>%  
 summarise(`Lower 2.5%` = quantile(value, 0.025),  
 `Upper 97.5%` = quantile(value, 0.975)) %>%  
 gather(key='Quantile', value='value', `Lower 2.5%`, `Upper 97.5%`)  
  
ggplot(hp.df, aes(x=value, fill=variable)) +  
 facet\_wrap(~variable, nrow=3, scales = "free") +  
 geom\_density(aes(y=..scaled..), alpha=0.5, color=NA) +   
 geom\_vline(data=hp.quantiles, aes(xintercept=value, color=Quantile)) +  
 labs(title="Posterior Regression Coefficients", x="Value", y="Density", fill="Variable")

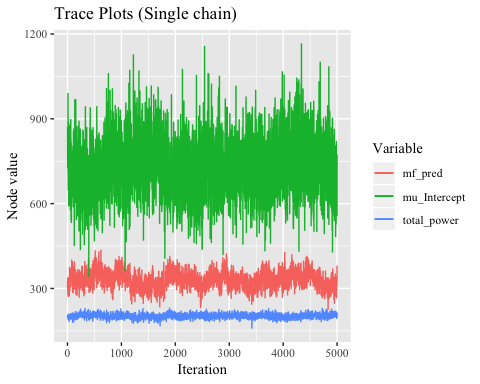


prod = as.data.frame(outmatrix) %>%  
 select(contains('prod')) %>%  
 gather(key=facility, value=value) %>%  
 mutate(which=parse\_number(facility)) %>%  
 mutate(whp=data$whp\_prod[which]) %>%  
 rename(mf=value) %>%  
 group\_by(whp) %>%  
 summarise(lower=quantile(mf, 0.025),  
 upper=quantile(mf, 0.975),  
 mean=mean(mf))  
  
plotdata = regression\_df %>%  
 filter(well\_id==ids[production\_curve\_well]) %>%  
 mutate(datetime = factor(as.Date(date))) %>%  
 # group\_by(datetime) %>%  
 # filter(n()>2) %>%  
 mutate(mf2 = mf^2)  
  
whp = seq(0, 16.5, 0.001)  
  
  
mylm = lm(mf ~ as.numeric(datetime) + whp, data=plotdata)  
datetime = seq(min(as.numeric(plotdata$datetime)), max(as.numeric(plotdata$datetime)), length.out=6)  
reglines = expand.grid(datetime=datetime, whp=whp)  
reglines$mf <- unname(predict(mylm, reglines))  
  
# mylm = lm(mf2 ~ as.numeric(datetime) \* I(whp^2) + as.numeric(datetime) , data=plotdata)  
# reglines = expand.grid(datetime=datetime, whp=whp)  
# reglines$mf2 <- unname(predict(mylm, reglines))  
# reglines = reglines %>% filter(mf2>=0)  
# reglines$mf <- sqrt(reglines$mf2)  
  
ggplot(prod, aes(x=whp)) +  
 geom\_line(aes(y=mean, lty="Bayesian Regression"), color='red') +  
 geom\_line(data=reglines, aes(y=mf, group=datetime, lty="OLS Regression")) +  
 geom\_ribbon(aes(ymin=lower, ymax=upper), alpha=0.25, fill='red') +  
 geom\_point(data=plotdata, aes(y=mf, color=date)) +  
 labs(title="Fitted Production Curve for WK255", x="Well-head pressure (bar)", y="Mass flow (T/d)", color="Date", linetype="") +  
 coord\_cartesian(xlim=c(8,16), ylim=c(0,750)) +  
 ggsave('../media/production\_curve.png', width=6, height=4, units='in')



# Trace plots

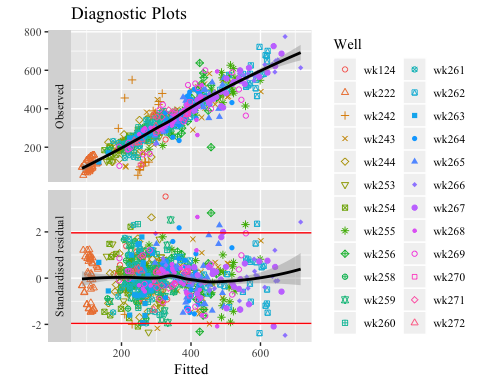
trace1 <- outframe %>%  
 filter(variable=='mf\_pred' & facility=='wk256')  
trace2 <- outframe %>%  
 filter(variable=='total\_power')  
trace3 <- outframe %>%  
 filter(variable=='mu\_Intercept')  
ggplot(trace1, aes( y=value, color=variable)) +  
 geom\_line(aes(x=as.numeric(row.names(trace1)))) +  
 geom\_line(data=trace2, aes(x=as.numeric(row.names(trace1)))) +  
 geom\_line(data=trace3, aes(x=as.numeric(row.names(trace1)))) +  
 labs(title="Trace Plots (Single chain)", x="Iteration", y="Node value", color="Variable") +  
 ggsave('../media/trace\_plots.png', width=6, height=4, units='in')



# Goodness of fit (OLS regression)

fit = as.data.frame(outmatrix) %>%  
 select(contains('mf\_fit')) %>%  
 gather(key='index', value='fitted') %>%  
 mutate(index=as.integer(parse\_number(index))) %>%  
 group\_by(index) %>%  
 summarise(lower=quantile(fitted, 0.025),  
 upper=quantile(fitted, 0.975),  
 Fitted=mean(fitted),  
 std=sd(fitted)) %>%  
 cbind(regression\_df) %>%  
 mutate(`Standardised residual` = (Fitted-mf)/std,  
 Well = factor(names(ids[well\_id])),  
 Observed = mf) %>%  
 gather(key="key", value="value", `Standardised residual`, Observed, )  
  
ggplot(fit, aes(x=Fitted, y=value)) +  
 geom\_point(aes(color=Well, shape=Well)) + scale\_shape\_manual(values = 1:length(levels(fit$Well))) +  
 geom\_smooth(color='black') +  
 facet\_grid(key~., scales="free\_y", switch="y") +  
 geom\_hline(data=data.frame(key="Standardised residual", value=c(1.96,-1.96)), aes(yintercept=value), color='red') +  
 labs(title="Diagnostic Plots", y="") +  
 ggsave('../media/diagnostics.png', width=6, height=4, units='in')

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



# Diagnostics

geweke.diag(out, 0.5, 0.5)

## [[1]]  
##   
## Fraction in 1st window = 0.5  
## Fraction in 2nd window = 0.5   
##   
## beta\_date[10] beta\_date[11] beta\_date[12] beta\_date[13] beta\_date[14]   
## 1.670029 0.893469 -0.679875 -0.015774 -1.946661   
## beta\_date[15] beta\_date[16] beta\_date[17] beta\_date[18] beta\_date[19]   
## 2.113587 -0.371665 -1.381797 1.189359 -0.561909   
## beta\_date[20] beta\_date[21] beta\_date[22] beta\_date[23] beta\_date[24]   
## 0.078086 1.053327 4.095025 -0.701719 -0.088319   
## beta\_date[25] beta\_date[2] beta\_date[3] beta\_date[4] beta\_date[5]   
## -0.406707 1.073887 1.279203 -0.646475 -0.943079   
## beta\_date[6] beta\_date[7] beta\_date[8] beta\_date[9] h[26]   
## 0.060835 -0.028587 -0.423204 1.382656 NaN   
## h[27] h[28] h[29] h[30] h[31]   
## NaN NaN NaN NaN -1.421169   
## h[32] h[33] h[34] h[35] h[36]   
## 0.594131 0.941816 NaN NaN NaN   
## h[37] mf\_fit[100] mf\_fit[101] mf\_fit[102] mf\_fit[103]   
## NaN -0.196696 -1.241824 0.290069 1.658409   
## mf\_fit[104] mf\_fit[105] mf\_fit[106] mf\_fit[107] mf\_fit[108]   
## 1.649871 0.780941 0.695946 1.449970 1.474866   
## mf\_fit[109] mf\_fit[10] mf\_fit[110] mf\_fit[111] mf\_fit[112]   
## 0.126677 -0.173387 0.464427 1.447009 0.352169   
## mf\_fit[113] mf\_fit[114] mf\_fit[115] mf\_fit[116] mf\_fit[117]   
## 0.357802 -0.792351 -0.817799 -0.507604 1.059386   
## mf\_fit[118] mf\_fit[119] mf\_fit[11] mf\_fit[120] mf\_fit[121]   
## 2.962192 -0.706360 -3.479434 0.964697 -0.230434   
## mf\_fit[122] mf\_fit[123] mf\_fit[124] mf\_fit[125] mf\_fit[126]   
## -0.944250 -2.403539 -1.656568 -0.904796 0.731963   
## mf\_fit[127] mf\_fit[128] mf\_fit[129] mf\_fit[12] mf\_fit[130]   
## -0.252895 0.351715 1.685686 -1.965808 -2.284392   
## mf\_fit[131] mf\_fit[132] mf\_fit[133] mf\_fit[134] mf\_fit[135]   
## -0.093275 -0.423289 -1.173932 -0.905536 0.815560   
## mf\_fit[136] mf\_fit[137] mf\_fit[138] mf\_fit[139] mf\_fit[13]   
## 0.229786 -1.384593 -0.222519 -2.391304 -0.074538   
## mf\_fit[140] mf\_fit[141] mf\_fit[142] mf\_fit[143] mf\_fit[144]   
## 1.781154 1.034228 -0.545836 -0.132240 -0.216185   
## mf\_fit[145] mf\_fit[146] mf\_fit[147] mf\_fit[148] mf\_fit[149]   
## -1.309306 -2.151018 -0.886949 -1.079961 -0.048855   
## mf\_fit[14] mf\_fit[150] mf\_fit[151] mf\_fit[152] mf\_fit[153]   
## -0.840476 0.650873 -0.533970 -1.827509 0.342832   
## mf\_fit[154] mf\_fit[155] mf\_fit[156] mf\_fit[157] mf\_fit[158]   
## 0.219965 -1.244931 -1.120025 0.549354 -0.425024   
## mf\_fit[159] mf\_fit[15] mf\_fit[160] mf\_fit[161] mf\_fit[162]   
## -0.619391 1.435447 -2.332859 -0.547893 -0.157539   
## mf\_fit[163] mf\_fit[164] mf\_fit[165] mf\_fit[166] mf\_fit[167]   
## -0.849402 -0.267963 -1.286788 -1.995126 1.002622   
## mf\_fit[168] mf\_fit[169] mf\_fit[16] mf\_fit[170] mf\_fit[171]   
## 0.651858 -0.867685 -1.147647 -0.642679 0.783187   
## mf\_fit[172] mf\_fit[173] mf\_fit[174] mf\_fit[175] mf\_fit[176]   
## -0.436913 -0.573100 0.862879 -0.229530 -0.616681   
## mf\_fit[177] mf\_fit[178] mf\_fit[179] mf\_fit[17] mf\_fit[180]   
## 0.118980 1.647998 -1.027608 -2.360043 1.214630   
## mf\_fit[181] mf\_fit[182] mf\_fit[183] mf\_fit[184] mf\_fit[185]   
## 0.325780 -0.561906 -1.452026 -0.948376 -0.809428   
## mf\_fit[186] mf\_fit[187] mf\_fit[188] mf\_fit[189] mf\_fit[18]   
## 0.768363 -0.426057 0.570279 -0.110004 -0.746656   
## mf\_fit[190] mf\_fit[191] mf\_fit[192] mf\_fit[193] mf\_fit[194]   
## -0.758926 0.424162 0.360882 1.455379 0.236024   
## mf\_fit[195] mf\_fit[196] mf\_fit[197] mf\_fit[198] mf\_fit[199]   
## -0.400246 -0.085357 0.248348 0.001877 -0.466168   
## mf\_fit[19] mf\_fit[1] mf\_fit[200] mf\_fit[201] mf\_fit[202]   
## -0.603723 -3.583596 -0.290061 -0.895158 -1.014284   
## mf\_fit[203] mf\_fit[204] mf\_fit[205] mf\_fit[206] mf\_fit[207]   
## -0.457192 -0.234183 0.622031 -0.297040 0.807931   
## mf\_fit[208] mf\_fit[209] mf\_fit[20] mf\_fit[210] mf\_fit[211]   
## -0.007383 0.069517 1.367867 -1.156780 -0.529798   
## mf\_fit[212] mf\_fit[213] mf\_fit[214] mf\_fit[215] mf\_fit[216]   
## -0.085372 -0.853255 -1.224988 -1.186919 0.448441   
## mf\_fit[217] mf\_fit[218] mf\_fit[219] mf\_fit[21] mf\_fit[220]   
## -0.377502 -0.199419 0.453803 3.571422 -2.140333   
## mf\_fit[221] mf\_fit[222] mf\_fit[223] mf\_fit[224] mf\_fit[225]   
## -0.182925 -1.398060 -0.354739 -0.809330 0.539037   
## mf\_fit[226] mf\_fit[227] mf\_fit[228] mf\_fit[229] mf\_fit[22]   
## -0.088467 0.195524 0.566290 0.265251 0.771224   
## mf\_fit[230] mf\_fit[231] mf\_fit[232] mf\_fit[233] mf\_fit[234]   
## -0.079932 0.037599 -0.250131 -0.017026 -0.910621   
## mf\_fit[235] mf\_fit[236] mf\_fit[237] mf\_fit[238] mf\_fit[239]   
## 0.762062 1.492113 -1.012685 1.156255 -2.417115   
## mf\_fit[23] mf\_fit[240] mf\_fit[241] mf\_fit[242] mf\_fit[243]   
## 0.961262 -1.612476 0.290967 -0.239884 -0.886759   
## mf\_fit[244] mf\_fit[245] mf\_fit[246] mf\_fit[247] mf\_fit[248]   
## -0.352282 0.030626 0.001044 -1.380424 -2.022264   
## mf\_fit[249] mf\_fit[24] mf\_fit[250] mf\_fit[251] mf\_fit[252]   
## 0.313939 0.069422 -1.065176 -1.385678 -1.244125   
## mf\_fit[253] mf\_fit[254] mf\_fit[255] mf\_fit[256] mf\_fit[257]   
## -0.256325 -1.279319 -1.788003 -0.533972 -0.384052   
## mf\_fit[258] mf\_fit[259] mf\_fit[25] mf\_fit[260] mf\_fit[261]   
## -1.114913 -1.031766 1.789473 -1.833525 0.405746   
## mf\_fit[262] mf\_fit[263] mf\_fit[264] mf\_fit[265] mf\_fit[266]   
## -1.267144 0.016276 -2.349650 -0.610481 -0.074760   
## mf\_fit[267] mf\_fit[268] mf\_fit[269] mf\_fit[26] mf\_fit[270]   
## -2.867417 -4.056621 -2.828429 3.243113 -3.276536   
## mf\_fit[271] mf\_fit[272] mf\_fit[273] mf\_fit[274] mf\_fit[275]   
## -3.693487 -0.953708 0.431936 -1.544816 2.969601   
## mf\_fit[276] mf\_fit[277] mf\_fit[278] mf\_fit[279] mf\_fit[27]   
## 2.705548 -0.165981 0.076015 -0.241269 1.501458   
## mf\_fit[280] mf\_fit[281] mf\_fit[282] mf\_fit[283] mf\_fit[284]   
## -1.977873 -0.340990 -3.676637 -1.883923 -2.877657   
## mf\_fit[285] mf\_fit[286] mf\_fit[287] mf\_fit[288] mf\_fit[289]   
## 1.172244 3.035935 2.678008 -1.801614 -2.673959   
## mf\_fit[28] mf\_fit[290] mf\_fit[291] mf\_fit[292] mf\_fit[293]   
## 2.160135 -2.637063 -1.844090 -0.827343 -0.792661   
## mf\_fit[294] mf\_fit[295] mf\_fit[296] mf\_fit[297] mf\_fit[298]   
## 4.870476 3.647203 2.380885 0.427525 -0.163959   
## mf\_fit[299] mf\_fit[29] mf\_fit[2] mf\_fit[300] mf\_fit[301]   
## -0.830680 3.782830 -3.546235 0.942021 2.033348   
## mf\_fit[302] mf\_fit[303] mf\_fit[304] mf\_fit[305] mf\_fit[306]   
## 3.176415 2.312168 0.375765 -1.446346 0.524080   
## mf\_fit[307] mf\_fit[308] mf\_fit[309] mf\_fit[30] mf\_fit[310]   
## -3.618199 -0.053363 1.513732 2.177201 -0.609550   
## mf\_fit[311] mf\_fit[312] mf\_fit[313] mf\_fit[314] mf\_fit[315]   
## 0.545074 1.705668 0.160690 -0.847648 -1.739834   
## mf\_fit[316] mf\_fit[317] mf\_fit[318] mf\_fit[319] mf\_fit[31]   
## -2.467775 -0.849822 1.508413 1.477869 2.042122   
## mf\_fit[320] mf\_fit[321] mf\_fit[322] mf\_fit[323] mf\_fit[324]   
## -0.514711 -0.740084 -1.397282 -0.390229 0.919367   
## mf\_fit[325] mf\_fit[326] mf\_fit[327] mf\_fit[328] mf\_fit[329]   
## -0.523122 1.213945 -0.008797 3.998683 4.768031   
## mf\_fit[32] mf\_fit[330] mf\_fit[331] mf\_fit[332] mf\_fit[333]   
## -1.233199 -2.422682 0.060459 -1.532724 3.022731   
## mf\_fit[334] mf\_fit[335] mf\_fit[336] mf\_fit[337] mf\_fit[338]   
## 1.930156 2.539615 1.808218 -1.595769 -1.016298   
## mf\_fit[339] mf\_fit[33] mf\_fit[340] mf\_fit[341] mf\_fit[342]   
## -0.785165 2.230637 -2.196861 -3.269168 -1.262526   
## mf\_fit[343] mf\_fit[344] mf\_fit[345] mf\_fit[346] mf\_fit[347]   
## -1.641945 -2.215909 -1.025134 0.254638 -2.642072   
## mf\_fit[348] mf\_fit[349] mf\_fit[34] mf\_fit[350] mf\_fit[351]   
## 0.974238 -0.348100 0.714960 -0.568422 -0.541609   
## mf\_fit[352] mf\_fit[353] mf\_fit[354] mf\_fit[355] mf\_fit[356]   
## 0.482590 0.736081 -0.071231 -3.028796 -0.577468   
## mf\_fit[357] mf\_fit[358] mf\_fit[359] mf\_fit[35] mf\_fit[360]   
## 1.831121 1.409459 0.038769 -0.643584 0.311867   
## mf\_fit[361] mf\_fit[362] mf\_fit[363] mf\_fit[364] mf\_fit[365]   
## 0.038023 1.526876 0.845008 1.504036 2.140352   
## mf\_fit[366] mf\_fit[367] mf\_fit[368] mf\_fit[369] mf\_fit[36]   
## 1.716637 2.047849 0.872154 4.032244 0.062989   
## mf\_fit[370] mf\_fit[371] mf\_fit[372] mf\_fit[373] mf\_fit[374]   
## 1.752960 0.491548 1.390897 0.682020 0.760638   
## mf\_fit[375] mf\_fit[376] mf\_fit[377] mf\_fit[378] mf\_fit[379]   
## 1.067607 1.722164 0.409192 0.383812 -0.689517   
## mf\_fit[37] mf\_fit[380] mf\_fit[381] mf\_fit[382] mf\_fit[383]   
## -0.855345 1.319955 -0.961791 3.066545 1.539063   
## mf\_fit[384] mf\_fit[385] mf\_fit[386] mf\_fit[387] mf\_fit[388]   
## -0.399234 1.152128 -0.052506 -1.220347 0.991924   
## mf\_fit[389] mf\_fit[38] mf\_fit[390] mf\_fit[391] mf\_fit[392]   
## -0.667951 -0.302763 -3.744275 -0.830922 2.352609   
## mf\_fit[393] mf\_fit[394] mf\_fit[395] mf\_fit[396] mf\_fit[397]   
## -1.453263 -1.174677 -0.675449 -0.157494 1.178710   
## mf\_fit[398] mf\_fit[399] mf\_fit[39] mf\_fit[3] mf\_fit[400]   
## -0.953374 -0.329837 -1.376956 -4.015148 -0.981637   
## mf\_fit[401] mf\_fit[402] mf\_fit[403] mf\_fit[404] mf\_fit[405]   
## -1.128143 -0.562571 -0.978079 -0.537666 -1.436960   
## mf\_fit[406] mf\_fit[407] mf\_fit[408] mf\_fit[409] mf\_fit[40]   
## -0.649289 0.313940 -1.676519 -1.120545 -1.025171   
## mf\_fit[410] mf\_fit[411] mf\_fit[412] mf\_fit[413] mf\_fit[414]   
## -1.068051 -1.661582 -0.183642 -0.267629 0.309820   
## mf\_fit[415] mf\_fit[416] mf\_fit[417] mf\_fit[418] mf\_fit[419]   
## 0.148193 0.442903 2.049508 1.204776 0.213693   
## mf\_fit[41] mf\_fit[420] mf\_fit[421] mf\_fit[422] mf\_fit[423]   
## -1.272005 0.258407 0.625038 0.802491 0.647511   
## mf\_fit[424] mf\_fit[425] mf\_fit[426] mf\_fit[427] mf\_fit[428]   
## -0.450666 2.027047 0.526433 0.991304 0.294330   
## mf\_fit[429] mf\_fit[42] mf\_fit[430] mf\_fit[431] mf\_fit[432]   
## 0.759879 -1.070819 0.959339 0.803517 -0.268341   
## mf\_fit[433] mf\_fit[434] mf\_fit[435] mf\_fit[436] mf\_fit[437]   
## 1.402862 1.607543 0.359143 0.407382 -0.273908   
## mf\_fit[438] mf\_fit[439] mf\_fit[43] mf\_fit[440] mf\_fit[441]   
## -1.939779 0.347576 -0.734759 0.007824 0.289972   
## mf\_fit[442] mf\_fit[443] mf\_fit[444] mf\_fit[445] mf\_fit[446]   
## 0.416194 -0.060795 0.011908 2.097787 2.352361   
## mf\_fit[447] mf\_fit[448] mf\_fit[449] mf\_fit[44] mf\_fit[450]   
## 3.798640 0.984542 3.352718 -3.314528 3.092178   
## mf\_fit[451] mf\_fit[452] mf\_fit[453] mf\_fit[454] mf\_fit[455]   
## 1.576432 1.890199 1.326433 3.179356 0.754133   
## mf\_fit[456] mf\_fit[457] mf\_fit[458] mf\_fit[459] mf\_fit[45]   
## -3.488847 0.898904 2.262195 0.522701 -1.187329   
## mf\_fit[460] mf\_fit[461] mf\_fit[462] mf\_fit[463] mf\_fit[464]   
## -1.385419 0.229966 0.766052 1.449371 1.497075   
## mf\_fit[465] mf\_fit[466] mf\_fit[467] mf\_fit[468] mf\_fit[469]   
## 0.058545 -1.936489 -1.127224 -1.951650 -0.099826   
## mf\_fit[46] mf\_fit[470] mf\_fit[471] mf\_fit[472] mf\_fit[473]   
## -1.094429 -0.680395 -0.692795 -1.986511 -1.221158   
## mf\_fit[474] mf\_fit[475] mf\_fit[476] mf\_fit[477] mf\_fit[478]   
## -2.367555 -3.433240 -1.930859 -2.532349 -2.385748   
## mf\_fit[479] mf\_fit[47] mf\_fit[480] mf\_fit[481] mf\_fit[482]   
## -2.670161 -0.999056 -2.412699 -2.572529 -2.873172   
## mf\_fit[483] mf\_fit[484] mf\_fit[485] mf\_fit[486] mf\_fit[487]   
## -0.380737 0.998143 1.496794 -1.806346 1.031480   
## mf\_fit[488] mf\_fit[489] mf\_fit[48] mf\_fit[490] mf\_fit[491]   
## -3.625151 -4.535326 -1.324515 1.365819 2.916308   
## mf\_fit[492] mf\_fit[493] mf\_fit[494] mf\_fit[495] mf\_fit[496]   
## -4.151636 -5.727583 -5.945264 -4.169970 -6.150881   
## mf\_fit[497] mf\_fit[498] mf\_fit[499] mf\_fit[49] mf\_fit[4]   
## -5.300947 -4.491229 0.920238 -1.510202 -2.775520   
## mf\_fit[500] mf\_fit[501] mf\_fit[502] mf\_fit[503] mf\_fit[504]   
## 2.133265 4.773782 -4.024407 -3.459905 -0.570251   
## mf\_fit[505] mf\_fit[506] mf\_fit[507] mf\_fit[508] mf\_fit[509]   
## -2.930474 -2.728864 5.378440 4.228875 6.442776   
## mf\_fit[50] mf\_fit[510] mf\_fit[511] mf\_fit[512] mf\_fit[513]   
## -2.686119 -1.336225 0.663514 2.074706 3.098036   
## mf\_fit[514] mf\_fit[515] mf\_fit[516] mf\_fit[517] mf\_fit[518]   
## 3.308426 4.917974 3.335322 2.864513 0.127382   
## mf\_fit[519] mf\_fit[51] mf\_fit[520] mf\_fit[521] mf\_fit[522]   
## 0.830625 -2.127052 0.714838 0.511401 -0.938185   
## mf\_fit[523] mf\_fit[524] mf\_fit[525] mf\_fit[526] mf\_fit[527]   
## -0.132398 1.151651 1.549162 -0.277289 0.765050   
## mf\_fit[528] mf\_fit[529] mf\_fit[52] mf\_fit[530] mf\_fit[531]   
## 1.306130 -0.698414 0.561399 -0.452684 -0.589180   
## mf\_fit[532] mf\_fit[533] mf\_fit[534] mf\_fit[535] mf\_fit[536]   
## -0.420942 -0.263925 -1.262073 -0.271681 1.279390   
## mf\_fit[537] mf\_fit[538] mf\_fit[539] mf\_fit[53] mf\_fit[540]   
## 0.339832 -2.178079 -0.098064 -0.027540 -1.655195   
## mf\_fit[541] mf\_fit[542] mf\_fit[543] mf\_fit[544] mf\_fit[545]   
## -1.433404 0.392869 -1.537817 1.405520 -0.172945   
## mf\_fit[546] mf\_fit[547] mf\_fit[548] mf\_fit[549] mf\_fit[54]   
## -1.658956 -0.705921 -1.113295 -1.524910 1.865137   
## mf\_fit[550] mf\_fit[551] mf\_fit[552] mf\_fit[553] mf\_fit[554]   
## -0.186005 -0.634898 1.419204 2.349497 2.964825   
## mf\_fit[555] mf\_fit[556] mf\_fit[557] mf\_fit[558] mf\_fit[559]   
## 1.750582 2.370469 3.358629 1.236755 1.337534   
## mf\_fit[55] mf\_fit[560] mf\_fit[561] mf\_fit[562] mf\_fit[563]   
## 0.621297 2.450084 2.822903 1.724592 1.490071   
## mf\_fit[564] mf\_fit[565] mf\_fit[566] mf\_fit[567] mf\_fit[568]   
## -1.354599 1.705229 -0.738914 -0.290530 1.524494   
## mf\_fit[569] mf\_fit[56] mf\_fit[570] mf\_fit[571] mf\_fit[572]   
## -1.751508 -0.543611 -1.765161 -1.079772 -0.849859   
## mf\_fit[573] mf\_fit[574] mf\_fit[575] mf\_fit[576] mf\_fit[577]   
## -0.530905 -0.850378 -4.962922 -1.757463 -1.185838   
## mf\_fit[578] mf\_fit[579] mf\_fit[57] mf\_fit[580] mf\_fit[581]   
## -0.917241 -0.598196 1.988906 -2.183991 -3.050355   
## mf\_fit[582] mf\_fit[583] mf\_fit[584] mf\_fit[585] mf\_fit[586]   
## -3.012261 -2.236944 -2.155700 -1.304357 -2.973272   
## mf\_fit[587] mf\_fit[588] mf\_fit[589] mf\_fit[58] mf\_fit[590]   
## -0.280238 -1.319876 -2.296472 2.164744 0.903798   
## mf\_fit[591] mf\_fit[592] mf\_fit[593] mf\_fit[594] mf\_fit[595]   
## -0.560043 -3.117853 -2.672938 -3.423780 -2.242606   
## mf\_fit[596] mf\_fit[597] mf\_fit[598] mf\_fit[599] mf\_fit[59]   
## -0.446029 2.250206 -2.360203 -2.513906 -0.510885   
## mf\_fit[5] mf\_fit[600] mf\_fit[601] mf\_fit[602] mf\_fit[603]   
## -1.683846 -2.060329 -1.868284 0.753427 0.637366   
## mf\_fit[604] mf\_fit[605] mf\_fit[606] mf\_fit[607] mf\_fit[608]   
## 1.528904 -2.382231 -0.530328 0.606604 0.371758   
## mf\_fit[609] mf\_fit[60] mf\_fit[610] mf\_fit[611] mf\_fit[612]   
## 2.485755 0.465692 4.410619 3.122342 0.217764   
## mf\_fit[613] mf\_fit[614] mf\_fit[615] mf\_fit[616] mf\_fit[617]   
## 1.042254 1.691126 0.198708 1.480171 1.196206   
## mf\_fit[618] mf\_fit[619] mf\_fit[61] mf\_fit[620] mf\_fit[621]   
## 0.698959 1.382616 3.023122 2.839985 0.056763   
## mf\_fit[622] mf\_fit[623] mf\_fit[624] mf\_fit[625] mf\_fit[626]   
## 0.565164 1.332214 2.379738 -0.527846 0.773290   
## mf\_fit[627] mf\_fit[628] mf\_fit[629] mf\_fit[62] mf\_fit[630]   
## 0.416698 1.395014 0.671996 0.321265 1.515818   
## mf\_fit[631] mf\_fit[632] mf\_fit[633] mf\_fit[634] mf\_fit[635]   
## 0.567515 1.039457 -1.005457 -0.079135 -1.536714   
## mf\_fit[636] mf\_fit[637] mf\_fit[638] mf\_fit[639] mf\_fit[63]   
## 0.195183 0.388820 1.965927 -0.543405 -0.634837   
## mf\_fit[640] mf\_fit[641] mf\_fit[642] mf\_fit[643] mf\_fit[644]   
## -0.411590 -1.425230 -0.924659 -0.996547 -0.700620   
## mf\_fit[645] mf\_fit[646] mf\_fit[647] mf\_fit[648] mf\_fit[649]   
## -0.976153 -0.485040 -1.324584 -1.211357 0.507974   
## mf\_fit[64] mf\_fit[650] mf\_fit[651] mf\_fit[652] mf\_fit[653]   
## 1.053679 -2.094481 0.080645 -0.394995 0.453571   
## mf\_fit[654] mf\_fit[655] mf\_fit[656] mf\_fit[657] mf\_fit[658]   
## -0.969348 -0.183830 -0.054936 -0.068524 -0.439907   
## mf\_fit[659] mf\_fit[65] mf\_fit[660] mf\_fit[661] mf\_fit[662]   
## -0.360973 0.567501 0.396361 0.435147 -0.171835   
## mf\_fit[663] mf\_fit[664] mf\_fit[665] mf\_fit[666] mf\_fit[667]   
## -0.589054 -0.527287 -1.020047 -0.321068 -2.705277   
## mf\_fit[668] mf\_fit[669] mf\_fit[66] mf\_fit[670] mf\_fit[671]   
## -0.348512 2.274784 1.486756 -1.278357 -0.104594   
## mf\_fit[672] mf\_fit[673] mf\_fit[674] mf\_fit[675] mf\_fit[676]   
## 0.861154 -1.065648 0.767233 -0.349539 0.117511   
## mf\_fit[677] mf\_fit[678] mf\_fit[679] mf\_fit[67] mf\_fit[680]   
## 0.831391 0.139091 1.838710 2.064784 -2.604547   
## mf\_fit[681] mf\_fit[682] mf\_fit[683] mf\_fit[684] mf\_fit[685]   
## 0.263693 -0.789274 1.091690 -0.547206 1.065690   
## mf\_fit[686] mf\_fit[687] mf\_fit[688] mf\_fit[689] mf\_fit[68]   
## 1.125250 -2.615003 -0.769896 -0.656576 1.551960   
## mf\_fit[690] mf\_fit[691] mf\_fit[692] mf\_fit[693] mf\_fit[694]   
## -0.295073 -0.250149 -2.323074 -0.686248 1.191182   
## mf\_fit[695] mf\_fit[696] mf\_fit[697] mf\_fit[698] mf\_fit[699]   
## 1.686775 0.144927 1.170257 -0.545309 1.253139   
## mf\_fit[69] mf\_fit[6] mf\_fit[700] mf\_fit[701] mf\_fit[702]   
## 0.666433 -0.320750 1.482219 0.436116 2.087964   
## mf\_fit[703] mf\_fit[704] mf\_fit[705] mf\_fit[706] mf\_fit[707]   
## 1.774152 1.447845 1.663948 -1.435948 -5.358834   
## mf\_fit[708] mf\_fit[709] mf\_fit[70] mf\_fit[710] mf\_fit[711]   
## -2.827146 -0.550424 1.124186 -1.225635 -1.079914   
## mf\_fit[712] mf\_fit[713] mf\_fit[714] mf\_fit[715] mf\_fit[716]   
## -2.007835 -3.138153 -0.629515 0.660949 -3.294940   
## mf\_fit[717] mf\_fit[718] mf\_fit[719] mf\_fit[71] mf\_fit[720]   
## -3.329091 -5.053916 -6.078621 -0.249157 -11.126587   
## mf\_fit[721] mf\_fit[722] mf\_fit[723] mf\_fit[724] mf\_fit[725]   
## 3.229790 5.348297 6.531410 5.155871 2.822635   
## mf\_fit[726] mf\_fit[727] mf\_fit[728] mf\_fit[729] mf\_fit[72]   
## 1.735433 1.806876 -0.406710 2.867093 -0.430023   
## mf\_fit[730] mf\_fit[731] mf\_fit[732] mf\_fit[733] mf\_fit[734]   
## 6.953837 4.536666 0.999239 -1.151933 -1.041818   
## mf\_fit[735] mf\_fit[736] mf\_fit[737] mf\_fit[738] mf\_fit[739]   
## 1.793132 -0.509179 0.050654 1.683949 0.905945   
## mf\_fit[73] mf\_fit[740] mf\_fit[741] mf\_fit[742] mf\_fit[743]   
## -0.598438 -0.542503 -0.018048 -0.661446 0.740392   
## mf\_fit[744] mf\_fit[745] mf\_fit[746] mf\_fit[747] mf\_fit[748]   
## -1.337623 -1.123262 0.367297 0.314353 0.756834   
## mf\_fit[749] mf\_fit[74] mf\_fit[75] mf\_fit[76] mf\_fit[77]   
## -0.270169 -0.798971 -0.326990 1.433503 1.072610   
## mf\_fit[78] mf\_fit[79] mf\_fit[7] mf\_fit[80] mf\_fit[81]   
## 1.590217 0.059456 -1.989027 -0.761236 0.290011   
## mf\_fit[82] mf\_fit[83] mf\_fit[84] mf\_fit[85] mf\_fit[86]   
## 2.006977 0.025839 -0.118498 2.093174 1.139309   
## mf\_fit[87] mf\_fit[88] mf\_fit[89] mf\_fit[8] mf\_fit[90]   
## 0.696054 1.624498 0.176129 -0.394599 -0.045859   
## mf\_fit[91] mf\_fit[92] mf\_fit[93] mf\_fit[94] mf\_fit[95]   
## -0.326296 1.329284 1.151158 0.354538 1.332733   
## mf\_fit[96] mf\_fit[97] mf\_fit[98] mf\_fit[99] mf\_fit[9]   
## -0.644501 -0.701784 1.405494 -1.388728 -0.251944   
## mf\_pred[10] mf\_pred[11] mf\_pred[12] mf\_pred[13] mf\_pred[14]   
## 1.712731 1.133297 -0.881024 0.090948 -2.319347   
## mf\_pred[15] mf\_pred[16] mf\_pred[17] mf\_pred[18] mf\_pred[19]   
## 2.232963 -0.527093 -1.476732 1.262295 -0.656682   
## mf\_pred[20] mf\_pred[21] mf\_pred[22] mf\_pred[23] mf\_pred[24]   
## 0.062517 1.036414 5.880921 -1.029867 0.431724   
## mf\_pred[25] mf\_pred[26] mf\_pred[27] mf\_pred[28] mf\_pred[29]   
## -0.595855 NaN NaN NaN 1.635593   
## mf\_pred[2] mf\_pred[30] mf\_pred[31] mf\_pred[32] mf\_pred[33]   
## 1.635593 NaN 1.675577 -0.483246 1.099267   
## mf\_pred[34] mf\_pred[35] mf\_pred[36] mf\_pred[37] mf\_pred[38]   
## NaN NaN NaN NaN NaN   
## mf\_pred[39] mf\_pred[3] mf\_pred[40] mf\_pred[41] mf\_pred[42]   
## NaN 1.240294 1.635593 1.114675 1.091263   
## mf\_pred[43] mf\_pred[44] mf\_pred[45] mf\_pred[46] mf\_pred[47]   
## NaN 1.264958 1.728733 NaN 0.019694   
## mf\_pred[48] mf\_pred[49] mf\_pred[4] mf\_pred[50] mf\_pred[51]   
## 0.577727 NaN -0.619485 1.635593 1.112063   
## mf\_pred[52] mf\_pred[53] mf\_pred[5] mf\_pred[6] mf\_pred[7]   
## 1.378195 0.081683 -1.083141 0.139043 -0.086860   
## mf\_pred[8] mf\_pred[9] mf\_prod[10] mf\_prod[1] mf\_prod[2]   
## -0.473638 2.934217 1.483693 -1.533574 -1.188733   
## mf\_prod[3] mf\_prod[4] mf\_prod[5] mf\_prod[6] mf\_prod[7]   
## 2.946206 1.889412 1.503177 1.439005 1.428885   
## mf\_prod[8] mf\_prod[9] mu\_Intercept mu\_beta\_date mu\_beta\_whp   
## 1.438663 1.461683 -0.868711 1.133457 4.351487   
## power[50] power[51] power[52] power[53] total\_power   
## 1.610905 1.214367 1.479876 -0.370642 1.257347   
##   
##   
## [[2]]  
##   
## Fraction in 1st window = 0.5  
## Fraction in 2nd window = 0.5   
##   
## beta\_date[10] beta\_date[11] beta\_date[12] beta\_date[13] beta\_date[14]   
## -0.5434046 0.5313042 0.4963565 0.3078848 1.6456427   
## beta\_date[15] beta\_date[16] beta\_date[17] beta\_date[18] beta\_date[19]   
## 1.5990497 -0.2289641 0.6240237 -0.6242108 1.2743330   
## beta\_date[20] beta\_date[21] beta\_date[22] beta\_date[23] beta\_date[24]   
## 1.2449849 -0.1662320 0.7038675 1.7122585 0.5912275   
## beta\_date[25] beta\_date[2] beta\_date[3] beta\_date[4] beta\_date[5]   
## 0.5913602 0.2044800 1.0924365 -2.3041500 -1.0217505   
## beta\_date[6] beta\_date[7] beta\_date[8] beta\_date[9] h[26]   
## 0.7660136 -0.1216467 -0.1862473 1.2389610 NaN   
## h[27] h[28] h[29] h[30] h[31]   
## NaN NaN NaN NaN 1.2400579   
## h[32] h[33] h[34] h[35] h[36]   
## 2.9675383 -0.2808564 NaN NaN NaN   
## h[37] mf\_fit[100] mf\_fit[101] mf\_fit[102] mf\_fit[103]   
## NaN -1.1020521 -1.4533210 -0.8676977 -1.6788347   
## mf\_fit[104] mf\_fit[105] mf\_fit[106] mf\_fit[107] mf\_fit[108]   
## 1.1975029 1.4034082 2.1679995 1.3144624 0.9647839   
## mf\_fit[109] mf\_fit[10] mf\_fit[110] mf\_fit[111] mf\_fit[112]   
## 1.5558853 0.5622304 -0.0446362 1.8650040 1.6710336   
## mf\_fit[113] mf\_fit[114] mf\_fit[115] mf\_fit[116] mf\_fit[117]   
## 1.0683797 -0.2917476 0.5231295 0.8772901 0.4407978   
## mf\_fit[118] mf\_fit[119] mf\_fit[11] mf\_fit[120] mf\_fit[121]   
## -0.3666297 0.1576094 -0.9888194 0.7977122 -0.7536877   
## mf\_fit[122] mf\_fit[123] mf\_fit[124] mf\_fit[125] mf\_fit[126]   
## -1.1865259 -0.7150964 -0.5325591 -0.6979570 1.5097855   
## mf\_fit[127] mf\_fit[128] mf\_fit[129] mf\_fit[12] mf\_fit[130]   
## 1.0703738 -0.2923591 1.6321741 -1.4498032 1.1307826   
## mf\_fit[131] mf\_fit[132] mf\_fit[133] mf\_fit[134] mf\_fit[135]   
## -0.4212247 -0.4443166 -0.2390147 -0.0855500 -1.0054091   
## mf\_fit[136] mf\_fit[137] mf\_fit[138] mf\_fit[139] mf\_fit[13]   
## 0.0370698 -0.0196008 -0.2692256 -0.0580863 0.3126350   
## mf\_fit[140] mf\_fit[141] mf\_fit[142] mf\_fit[143] mf\_fit[144]   
## -1.4714174 -1.8272698 -0.6474778 1.7447180 -0.3487824   
## mf\_fit[145] mf\_fit[146] mf\_fit[147] mf\_fit[148] mf\_fit[149]   
## 0.0161524 -1.6597906 -1.0513041 -1.7037229 -0.6655475   
## mf\_fit[14] mf\_fit[150] mf\_fit[151] mf\_fit[152] mf\_fit[153]   
## -2.0128844 -1.7681495 -0.9849667 -1.1080210 -2.3527572   
## mf\_fit[154] mf\_fit[155] mf\_fit[156] mf\_fit[157] mf\_fit[158]   
## -2.1245060 -2.4018470 0.9987711 0.8543581 -0.7385223   
## mf\_fit[159] mf\_fit[15] mf\_fit[160] mf\_fit[161] mf\_fit[162]   
## -1.6067960 0.1633571 -1.4388083 -1.1080132 0.4567290   
## mf\_fit[163] mf\_fit[164] mf\_fit[165] mf\_fit[166] mf\_fit[167]   
## -0.3088284 -1.9732018 -0.3361255 -0.7059604 -2.0447138   
## mf\_fit[168] mf\_fit[169] mf\_fit[16] mf\_fit[170] mf\_fit[171]   
## -0.6407568 0.9945151 -1.6831587 -0.4143276 -0.3611533   
## mf\_fit[172] mf\_fit[173] mf\_fit[174] mf\_fit[175] mf\_fit[176]   
## 1.8967636 0.3255241 -1.6912420 -0.7850061 -0.8998778   
## mf\_fit[177] mf\_fit[178] mf\_fit[179] mf\_fit[17] mf\_fit[180]   
## -2.2352106 -0.2371070 0.4033474 0.1493534 1.3037960   
## mf\_fit[181] mf\_fit[182] mf\_fit[183] mf\_fit[184] mf\_fit[185]   
## -0.8668725 -0.1546246 0.9784960 0.0048851 0.3418248   
## mf\_fit[186] mf\_fit[187] mf\_fit[188] mf\_fit[189] mf\_fit[18]   
## 2.1701602 -0.7520584 -0.0612382 0.4120710 -1.8340752   
## mf\_fit[190] mf\_fit[191] mf\_fit[192] mf\_fit[193] mf\_fit[194]   
## 0.2710200 1.1674972 0.2439239 -1.0543866 0.2408600   
## mf\_fit[195] mf\_fit[196] mf\_fit[197] mf\_fit[198] mf\_fit[199]   
## -0.1779154 -2.6000313 -0.2990573 0.3266893 0.8768534   
## mf\_fit[19] mf\_fit[1] mf\_fit[200] mf\_fit[201] mf\_fit[202]   
## -1.0325490 0.3473540 0.4632323 0.1857018 0.2787489   
## mf\_fit[203] mf\_fit[204] mf\_fit[205] mf\_fit[206] mf\_fit[207]   
## -1.1450429 -1.9336070 -0.9907969 0.6389714 -0.6203685   
## mf\_fit[208] mf\_fit[209] mf\_fit[20] mf\_fit[210] mf\_fit[211]   
## 0.0635805 -1.4379647 0.9306329 0.7984587 -0.1926677   
## mf\_fit[212] mf\_fit[213] mf\_fit[214] mf\_fit[215] mf\_fit[216]   
## -0.9242697 -0.9944196 -0.8910304 1.0153717 0.1332451   
## mf\_fit[217] mf\_fit[218] mf\_fit[219] mf\_fit[21] mf\_fit[220]   
## -0.7809473 0.5273458 -1.7622924 -0.8746385 -0.6911882   
## mf\_fit[221] mf\_fit[222] mf\_fit[223] mf\_fit[224] mf\_fit[225]   
## -1.4754526 -1.1705691 -0.0154771 -1.4035583 0.0865683   
## mf\_fit[226] mf\_fit[227] mf\_fit[228] mf\_fit[229] mf\_fit[22]   
## 0.1664608 -0.8606042 0.1033247 0.0915809 -0.0426773   
## mf\_fit[230] mf\_fit[231] mf\_fit[232] mf\_fit[233] mf\_fit[234]   
## -0.6162716 0.8103030 -0.0304798 1.1422279 0.5982228   
## mf\_fit[235] mf\_fit[236] mf\_fit[237] mf\_fit[238] mf\_fit[239]   
## 1.0749237 0.3760479 -1.2769072 1.7948514 0.3620025   
## mf\_fit[23] mf\_fit[240] mf\_fit[241] mf\_fit[242] mf\_fit[243]   
## 0.2691746 0.7684058 0.9780413 0.0383616 -0.9013822   
## mf\_fit[244] mf\_fit[245] mf\_fit[246] mf\_fit[247] mf\_fit[248]   
## -0.8869313 0.5299275 0.8367426 0.0174830 -0.6414290   
## mf\_fit[249] mf\_fit[24] mf\_fit[250] mf\_fit[251] mf\_fit[252]   
## -0.7297621 0.0325137 0.4348856 0.5869230 -0.6969564   
## mf\_fit[253] mf\_fit[254] mf\_fit[255] mf\_fit[256] mf\_fit[257]   
## 0.9579514 1.1215099 0.1763744 0.1989170 1.1212256   
## mf\_fit[258] mf\_fit[259] mf\_fit[25] mf\_fit[260] mf\_fit[261]   
## 0.1909210 0.6587879 -0.7596060 0.5401309 0.9057877   
## mf\_fit[262] mf\_fit[263] mf\_fit[264] mf\_fit[265] mf\_fit[266]   
## 0.2803627 -0.1829476 -2.1989239 -1.6605797 -1.2888998   
## mf\_fit[267] mf\_fit[268] mf\_fit[269] mf\_fit[26] mf\_fit[270]   
## -2.6510882 -2.5283704 -1.2299680 0.2245388 -1.6201839   
## mf\_fit[271] mf\_fit[272] mf\_fit[273] mf\_fit[274] mf\_fit[275]   
## -1.4521691 -0.9981296 3.2486797 -2.2744742 3.1027779   
## mf\_fit[276] mf\_fit[277] mf\_fit[278] mf\_fit[279] mf\_fit[27]   
## 3.6514648 1.2686838 -2.0620159 0.0011562 0.0453767   
## mf\_fit[280] mf\_fit[281] mf\_fit[282] mf\_fit[283] mf\_fit[284]   
## -1.4259513 -0.6642091 -4.0676440 -2.7094196 -1.5771082   
## mf\_fit[285] mf\_fit[286] mf\_fit[287] mf\_fit[288] mf\_fit[289]   
## 1.8777730 4.1378062 2.0206005 -2.8950377 -1.1840059   
## mf\_fit[28] mf\_fit[290] mf\_fit[291] mf\_fit[292] mf\_fit[293]   
## 0.5931595 -1.3605578 -0.7859286 -0.7752955 -0.6649524   
## mf\_fit[294] mf\_fit[295] mf\_fit[296] mf\_fit[297] mf\_fit[298]   
## 3.7840745 3.2467743 1.6061123 1.6241409 0.4030443   
## mf\_fit[299] mf\_fit[29] mf\_fit[2] mf\_fit[300] mf\_fit[301]   
## 1.2454924 1.3373549 0.6524240 0.2392313 2.3813269   
## mf\_fit[302] mf\_fit[303] mf\_fit[304] mf\_fit[305] mf\_fit[306]   
## 3.0668626 2.3705682 -1.0269711 0.6424624 0.1915772   
## mf\_fit[307] mf\_fit[308] mf\_fit[309] mf\_fit[30] mf\_fit[310]   
## 0.1260990 0.2651887 -0.8801103 0.7856196 0.2481790   
## mf\_fit[311] mf\_fit[312] mf\_fit[313] mf\_fit[314] mf\_fit[315]   
## -0.5095410 -0.3157037 -0.0487183 0.0010683 0.6555571   
## mf\_fit[316] mf\_fit[317] mf\_fit[318] mf\_fit[319] mf\_fit[31]   
## 0.5006088 -1.2632681 1.0436307 0.0468868 0.7373103   
## mf\_fit[320] mf\_fit[321] mf\_fit[322] mf\_fit[323] mf\_fit[324]   
## -0.9685268 -0.4435262 0.6512734 -0.4275306 0.3282667   
## mf\_fit[325] mf\_fit[326] mf\_fit[327] mf\_fit[328] mf\_fit[329]   
## 1.5191072 0.0076692 -0.3426040 -0.3356418 -2.3050184   
## mf\_fit[32] mf\_fit[330] mf\_fit[331] mf\_fit[332] mf\_fit[333]   
## -0.1358545 -0.8814114 0.6542989 0.9601824 -1.4561535   
## mf\_fit[334] mf\_fit[335] mf\_fit[336] mf\_fit[337] mf\_fit[338]   
## -0.7398323 -0.6724022 -1.4074601 -0.3944643 -1.8301063   
## mf\_fit[339] mf\_fit[33] mf\_fit[340] mf\_fit[341] mf\_fit[342]   
## -0.6306699 0.6627707 -1.2311432 -2.8120453 -1.8391248   
## mf\_fit[343] mf\_fit[344] mf\_fit[345] mf\_fit[346] mf\_fit[347]   
## -0.6792787 -0.6926104 -1.0529352 -1.5257757 -0.8590514   
## mf\_fit[348] mf\_fit[349] mf\_fit[34] mf\_fit[350] mf\_fit[351]   
## 0.3568645 0.0496084 0.6586744 0.5815355 -0.2969873   
## mf\_fit[352] mf\_fit[353] mf\_fit[354] mf\_fit[355] mf\_fit[356]   
## -0.2616686 1.6411091 0.6905820 -0.9051492 -1.3242311   
## mf\_fit[357] mf\_fit[358] mf\_fit[359] mf\_fit[35] mf\_fit[360]   
## -0.1365995 -1.1374198 -0.3087699 0.1746370 1.7455898   
## mf\_fit[361] mf\_fit[362] mf\_fit[363] mf\_fit[364] mf\_fit[365]   
## -1.0585676 1.7806777 1.0577451 -0.6070287 -0.1633731   
## mf\_fit[366] mf\_fit[367] mf\_fit[368] mf\_fit[369] mf\_fit[36]   
## -0.5260696 0.1522925 2.1428001 0.5259782 -0.2343664   
## mf\_fit[370] mf\_fit[371] mf\_fit[372] mf\_fit[373] mf\_fit[374]   
## 2.3380036 0.9225521 -1.4908336 -0.2820097 2.5382117   
## mf\_fit[375] mf\_fit[376] mf\_fit[377] mf\_fit[378] mf\_fit[379]   
## 0.2094287 -0.0709833 0.0366807 -1.3319273 -0.2033615   
## mf\_fit[37] mf\_fit[380] mf\_fit[381] mf\_fit[382] mf\_fit[383]   
## -0.4212000 0.3166929 -0.3974838 -0.9304779 -2.2450770   
## mf\_fit[384] mf\_fit[385] mf\_fit[386] mf\_fit[387] mf\_fit[388]   
## 0.2067447 -0.9676808 -1.4021063 0.9783952 -0.5411459   
## mf\_fit[389] mf\_fit[38] mf\_fit[390] mf\_fit[391] mf\_fit[392]   
## 0.2307572 -0.1630090 0.6935248 -0.0820453 -0.7014359   
## mf\_fit[393] mf\_fit[394] mf\_fit[395] mf\_fit[396] mf\_fit[397]   
## 0.0007796 0.8856773 -1.0245406 -1.4076522 -1.1873402   
## mf\_fit[398] mf\_fit[399] mf\_fit[39] mf\_fit[3] mf\_fit[400]   
## 0.1123656 -0.2413434 -1.1726577 -1.5220684 0.0395972   
## mf\_fit[401] mf\_fit[402] mf\_fit[403] mf\_fit[404] mf\_fit[405]   
## 0.0131563 -0.6671997 1.1730944 1.3959122 -0.8931229   
## mf\_fit[406] mf\_fit[407] mf\_fit[408] mf\_fit[409] mf\_fit[40]   
## 1.2272298 0.3467668 0.2240825 1.0066656 -0.6397996   
## mf\_fit[410] mf\_fit[411] mf\_fit[412] mf\_fit[413] mf\_fit[414]   
## 0.4366061 0.7121289 -0.2016493 -0.0632226 -0.5230049   
## mf\_fit[415] mf\_fit[416] mf\_fit[417] mf\_fit[418] mf\_fit[419]   
## -1.7324993 0.1098260 0.1392870 0.5734266 -1.7247956   
## mf\_fit[41] mf\_fit[420] mf\_fit[421] mf\_fit[422] mf\_fit[423]   
## -1.0572394 0.3003547 -0.5536193 -0.9792400 2.8143426   
## mf\_fit[424] mf\_fit[425] mf\_fit[426] mf\_fit[427] mf\_fit[428]   
## -0.8233498 0.1300400 -0.9538646 0.4628935 0.9634659   
## mf\_fit[429] mf\_fit[42] mf\_fit[430] mf\_fit[431] mf\_fit[432]   
## -0.2882973 -0.6139675 0.5319181 1.2589335 -0.0031022   
## mf\_fit[433] mf\_fit[434] mf\_fit[435] mf\_fit[436] mf\_fit[437]   
## 0.3116367 0.9144955 0.3373036 2.0967704 0.5352755   
## mf\_fit[438] mf\_fit[439] mf\_fit[43] mf\_fit[440] mf\_fit[441]   
## -0.6418456 1.6757730 -0.8309999 -0.8742729 -0.1671001   
## mf\_fit[442] mf\_fit[443] mf\_fit[444] mf\_fit[445] mf\_fit[446]   
## 1.0777133 0.1565890 0.8392424 -1.7677126 -2.3118922   
## mf\_fit[447] mf\_fit[448] mf\_fit[449] mf\_fit[44] mf\_fit[450]   
## -2.1822202 -2.0573098 -3.4772094 -1.5886771 -2.7441338   
## mf\_fit[451] mf\_fit[452] mf\_fit[453] mf\_fit[454] mf\_fit[455]   
## -2.5942033 2.2508647 -0.8817129 -2.5825792 -2.4107622   
## mf\_fit[456] mf\_fit[457] mf\_fit[458] mf\_fit[459] mf\_fit[45]   
## 0.3087764 -1.8043282 -0.0669714 -1.3599256 -1.9273355   
## mf\_fit[460] mf\_fit[461] mf\_fit[462] mf\_fit[463] mf\_fit[464]   
## -0.9764168 -1.0191729 -0.2811966 0.4753756 -0.0949579   
## mf\_fit[465] mf\_fit[466] mf\_fit[467] mf\_fit[468] mf\_fit[469]   
## 0.2636716 1.9943409 0.2112639 0.6504768 1.5941158   
## mf\_fit[46] mf\_fit[470] mf\_fit[471] mf\_fit[472] mf\_fit[473]   
## -1.0905967 0.3504855 0.8284336 2.0629567 2.0498452   
## mf\_fit[474] mf\_fit[475] mf\_fit[476] mf\_fit[477] mf\_fit[478]   
## 1.2901679 2.8981808 3.0407802 2.2737941 3.3103571   
## mf\_fit[479] mf\_fit[47] mf\_fit[480] mf\_fit[481] mf\_fit[482]   
## 1.4752805 -3.0107457 2.7173498 2.5961741 -2.9366032   
## mf\_fit[483] mf\_fit[484] mf\_fit[485] mf\_fit[486] mf\_fit[487]   
## -0.3326139 0.0401290 0.3594157 -0.9391881 -0.2629796   
## mf\_fit[488] mf\_fit[489] mf\_fit[48] mf\_fit[490] mf\_fit[491]   
## -2.7291519 -4.0524973 -1.5617517 0.5783846 1.7233058   
## mf\_fit[492] mf\_fit[493] mf\_fit[494] mf\_fit[495] mf\_fit[496]   
## -4.9624849 -3.0466250 -5.3423500 -4.1879567 -2.5631939   
## mf\_fit[497] mf\_fit[498] mf\_fit[499] mf\_fit[49] mf\_fit[4]   
## -2.1563384 -2.3792817 0.2346621 -1.3115846 -0.5055343   
## mf\_fit[500] mf\_fit[501] mf\_fit[502] mf\_fit[503] mf\_fit[504]   
## 1.8218646 2.1682994 -3.6369497 -1.6464544 -0.3584296   
## mf\_fit[505] mf\_fit[506] mf\_fit[507] mf\_fit[508] mf\_fit[509]   
## -2.1486256 -1.4288285 2.7367426 4.6532692 3.5326675   
## mf\_fit[50] mf\_fit[510] mf\_fit[511] mf\_fit[512] mf\_fit[513]   
## -0.9252205 -0.3987818 -1.0927124 0.2606906 3.1276254   
## mf\_fit[514] mf\_fit[515] mf\_fit[516] mf\_fit[517] mf\_fit[518]   
## 1.9224981 3.7115314 1.6676275 2.6147948 0.7284180   
## mf\_fit[519] mf\_fit[51] mf\_fit[520] mf\_fit[521] mf\_fit[522]   
## -0.3047320 -1.2321096 -0.1266909 -0.8085234 0.0995731   
## mf\_fit[523] mf\_fit[524] mf\_fit[525] mf\_fit[526] mf\_fit[527]   
## 1.4004345 0.5094735 1.1964463 -0.2483894 -0.0726818   
## mf\_fit[528] mf\_fit[529] mf\_fit[52] mf\_fit[530] mf\_fit[531]   
## -0.4448758 -0.1589693 0.4829998 -0.3398759 0.7602396   
## mf\_fit[532] mf\_fit[533] mf\_fit[534] mf\_fit[535] mf\_fit[536]   
## -1.2213126 -0.4673002 0.5872979 1.3772699 -1.0825363   
## mf\_fit[537] mf\_fit[538] mf\_fit[539] mf\_fit[53] mf\_fit[540]   
## -0.0777759 1.2877990 -2.3803252 -0.3814064 -1.1904296   
## mf\_fit[541] mf\_fit[542] mf\_fit[543] mf\_fit[544] mf\_fit[545]   
## -1.7842478 0.3899113 -2.5537181 0.3854780 -1.4242663   
## mf\_fit[546] mf\_fit[547] mf\_fit[548] mf\_fit[549] mf\_fit[54]   
## 0.4591653 -0.8995062 0.1460670 0.0050923 0.6875245   
## mf\_fit[550] mf\_fit[551] mf\_fit[552] mf\_fit[553] mf\_fit[554]   
## 0.6507425 0.5977886 -0.4234904 -0.6588329 -0.5761040   
## mf\_fit[555] mf\_fit[556] mf\_fit[557] mf\_fit[558] mf\_fit[559]   
## 0.0530693 -0.9500946 0.4616563 0.3664624 0.0644918   
## mf\_fit[55] mf\_fit[560] mf\_fit[561] mf\_fit[562] mf\_fit[563]   
## -0.5513153 -0.1578261 0.1155230 -2.3575962 -0.9902866   
## mf\_fit[564] mf\_fit[565] mf\_fit[566] mf\_fit[567] mf\_fit[568]   
## -0.8669995 -0.3840660 -0.0306086 -0.1084169 -1.7624083   
## mf\_fit[569] mf\_fit[56] mf\_fit[570] mf\_fit[571] mf\_fit[572]   
## 0.0471846 0.6399752 -0.0186126 -1.5397309 0.1683733   
## mf\_fit[573] mf\_fit[574] mf\_fit[575] mf\_fit[576] mf\_fit[577]   
## -1.3709277 2.4646520 0.1019673 0.3912827 0.5528123   
## mf\_fit[578] mf\_fit[579] mf\_fit[57] mf\_fit[580] mf\_fit[581]   
## 1.6177146 1.0166169 -2.5767437 0.4404325 0.1456270   
## mf\_fit[582] mf\_fit[583] mf\_fit[584] mf\_fit[585] mf\_fit[586]   
## -0.6318447 0.7290949 0.5073830 1.3560400 2.1076443   
## mf\_fit[587] mf\_fit[588] mf\_fit[589] mf\_fit[58] mf\_fit[590]   
## 2.9161328 2.3288324 0.4939430 0.7352767 -0.1135255   
## mf\_fit[591] mf\_fit[592] mf\_fit[593] mf\_fit[594] mf\_fit[595]   
## -0.7893818 0.9288649 0.8287690 2.4502212 0.4837088   
## mf\_fit[596] mf\_fit[597] mf\_fit[598] mf\_fit[599] mf\_fit[59]   
## 0.7373401 0.5689284 -0.2842670 1.8236151 0.3294970   
## mf\_fit[5] mf\_fit[600] mf\_fit[601] mf\_fit[602] mf\_fit[603]   
## -0.4216926 1.1976385 -0.2912374 0.9821245 0.6241435   
## mf\_fit[604] mf\_fit[605] mf\_fit[606] mf\_fit[607] mf\_fit[608]   
## -1.2858382 0.7162010 2.2838863 -0.7429708 0.8488329   
## mf\_fit[609] mf\_fit[60] mf\_fit[610] mf\_fit[611] mf\_fit[612]   
## -0.9521942 1.5764206 1.2751506 -1.9105249 -1.2390916   
## mf\_fit[613] mf\_fit[614] mf\_fit[615] mf\_fit[616] mf\_fit[617]   
## 1.9490621 -1.8044184 -0.3603919 -0.6817204 -0.7982730   
## mf\_fit[618] mf\_fit[619] mf\_fit[61] mf\_fit[620] mf\_fit[621]   
## -0.4673283 -0.2691990 0.4555972 -1.0141284 -1.4106762   
## mf\_fit[622] mf\_fit[623] mf\_fit[624] mf\_fit[625] mf\_fit[626]   
## -1.9088535 0.3340944 -0.5921129 -0.2736330 -0.6844752   
## mf\_fit[627] mf\_fit[628] mf\_fit[629] mf\_fit[62] mf\_fit[630]   
## -0.8378413 -2.7126352 -1.3642900 2.0339971 -2.4510896   
## mf\_fit[631] mf\_fit[632] mf\_fit[633] mf\_fit[634] mf\_fit[635]   
## -2.3386398 -1.9790507 2.9133895 1.0390974 0.9550441   
## mf\_fit[636] mf\_fit[637] mf\_fit[638] mf\_fit[639] mf\_fit[63]   
## -1.9369166 -0.5736917 -0.3791878 0.5851087 1.3856101   
## mf\_fit[640] mf\_fit[641] mf\_fit[642] mf\_fit[643] mf\_fit[644]   
## 1.3200259 1.4458245 2.6696899 2.1910273 1.6140510   
## mf\_fit[645] mf\_fit[646] mf\_fit[647] mf\_fit[648] mf\_fit[649]   
## 1.8272828 1.9696002 0.1876502 3.0312003 -1.5819617   
## mf\_fit[64] mf\_fit[650] mf\_fit[651] mf\_fit[652] mf\_fit[653]   
## -0.5190592 -2.4932009 -3.9480513 -2.4693489 -2.3688077   
## mf\_fit[654] mf\_fit[655] mf\_fit[656] mf\_fit[657] mf\_fit[658]   
## -3.7277887 -2.1399666 -2.9003207 -2.8477292 -4.0914497   
## mf\_fit[659] mf\_fit[65] mf\_fit[660] mf\_fit[661] mf\_fit[662]   
## 1.1129007 0.2215190 -0.7350425 0.5338402 -0.1775189   
## mf\_fit[663] mf\_fit[664] mf\_fit[665] mf\_fit[666] mf\_fit[667]   
## 1.4320912 0.0223066 0.3347097 -1.2854107 -3.2095356   
## mf\_fit[668] mf\_fit[669] mf\_fit[66] mf\_fit[670] mf\_fit[671]   
## -0.5631043 2.7891670 0.1310703 2.9235129 2.7625321   
## mf\_fit[672] mf\_fit[673] mf\_fit[674] mf\_fit[675] mf\_fit[676]   
## 1.2130470 0.5279231 3.1076449 0.5387219 2.3622412   
## mf\_fit[677] mf\_fit[678] mf\_fit[679] mf\_fit[67] mf\_fit[680]   
## 2.2300117 1.3555324 3.0637028 0.7912688 0.2534416   
## mf\_fit[681] mf\_fit[682] mf\_fit[683] mf\_fit[684] mf\_fit[685]   
## -0.3714530 -0.1824522 -2.5567748 -0.3272863 0.4067715   
## mf\_fit[686] mf\_fit[687] mf\_fit[688] mf\_fit[689] mf\_fit[68]   
## -0.1117962 -0.6735531 -1.1558534 1.0336046 -0.6379547   
## mf\_fit[690] mf\_fit[691] mf\_fit[692] mf\_fit[693] mf\_fit[694]   
## -0.3636929 -0.3508721 1.0575294 1.6319284 1.0253378   
## mf\_fit[695] mf\_fit[696] mf\_fit[697] mf\_fit[698] mf\_fit[699]   
## 0.5434919 0.2630869 -0.5304909 -0.6278639 0.4014273   
## mf\_fit[69] mf\_fit[6] mf\_fit[700] mf\_fit[701] mf\_fit[702]   
## -0.1518732 -0.3217908 0.5633661 0.6683427 0.7746005   
## mf\_fit[703] mf\_fit[704] mf\_fit[705] mf\_fit[706] mf\_fit[707]   
## -0.1326545 -0.1461538 -0.2407316 0.9987161 -0.8787341   
## mf\_fit[708] mf\_fit[709] mf\_fit[70] mf\_fit[710] mf\_fit[711]   
## -1.1677980 -0.5048569 -1.2173832 -0.4689163 -0.5219438   
## mf\_fit[712] mf\_fit[713] mf\_fit[714] mf\_fit[715] mf\_fit[716]   
## 1.2168150 -0.3009306 0.0049314 1.6541993 -1.2714735   
## mf\_fit[717] mf\_fit[718] mf\_fit[719] mf\_fit[71] mf\_fit[720]   
## -0.9082284 -2.5285420 -1.7953383 -1.8207442 -1.3412622   
## mf\_fit[721] mf\_fit[722] mf\_fit[723] mf\_fit[724] mf\_fit[725]   
## -0.0672683 1.6995151 2.2108478 0.8933951 1.1313431   
## mf\_fit[726] mf\_fit[727] mf\_fit[728] mf\_fit[729] mf\_fit[72]   
## 0.9137129 -0.4359998 -0.3888917 1.0727130 -2.3963560   
## mf\_fit[730] mf\_fit[731] mf\_fit[732] mf\_fit[733] mf\_fit[734]   
## 1.8069026 -0.1118983 -1.8866059 0.8378685 0.7317698   
## mf\_fit[735] mf\_fit[736] mf\_fit[737] mf\_fit[738] mf\_fit[739]   
## 1.2728234 -1.1161442 -0.0839945 -0.1935888 -0.1653068   
## mf\_fit[73] mf\_fit[740] mf\_fit[741] mf\_fit[742] mf\_fit[743]   
## 0.4623897 0.2476435 -0.5272408 0.4681197 -1.6724097   
## mf\_fit[744] mf\_fit[745] mf\_fit[746] mf\_fit[747] mf\_fit[748]   
## -0.9409369 0.6145816 -0.1264903 -0.5680008 2.7162568   
## mf\_fit[749] mf\_fit[74] mf\_fit[75] mf\_fit[76] mf\_fit[77]   
## -0.1919516 -0.5122337 1.3338671 2.0830097 0.6538143   
## mf\_fit[78] mf\_fit[79] mf\_fit[7] mf\_fit[80] mf\_fit[81]   
## 0.7410372 2.0416753 -0.7617033 1.4231989 1.0309603   
## mf\_fit[82] mf\_fit[83] mf\_fit[84] mf\_fit[85] mf\_fit[86]   
## 0.2265340 0.8427650 0.4165924 1.6701817 -0.3023482   
## mf\_fit[87] mf\_fit[88] mf\_fit[89] mf\_fit[8] mf\_fit[90]   
## 1.0528028 -0.2817834 0.6216552 1.1603951 0.7350118   
## mf\_fit[91] mf\_fit[92] mf\_fit[93] mf\_fit[94] mf\_fit[95]   
## -0.9173613 0.5103837 1.2797006 -2.0165680 -0.3887310   
## mf\_fit[96] mf\_fit[97] mf\_fit[98] mf\_fit[99] mf\_fit[9]   
## -2.7060550 -2.8482641 -2.7580631 -1.6910272 -0.0828277   
## mf\_pred[10] mf\_pred[11] mf\_pred[12] mf\_pred[13] mf\_pred[14]   
## -0.5903864 0.9986749 0.5057963 0.2818571 1.9677647   
## mf\_pred[15] mf\_pred[16] mf\_pred[17] mf\_pred[18] mf\_pred[19]   
## 1.8692702 -0.1571383 0.7800440 -0.5454797 1.7736444   
## mf\_pred[20] mf\_pred[21] mf\_pred[22] mf\_pred[23] mf\_pred[24]   
## 1.5761573 -0.2049747 1.0089096 2.1414127 0.4717082   
## mf\_pred[25] mf\_pred[26] mf\_pred[27] mf\_pred[28] mf\_pred[29]   
## 0.8405213 NaN NaN NaN 0.2707852   
## mf\_pred[2] mf\_pred[30] mf\_pred[31] mf\_pred[32] mf\_pred[33]   
## 0.2707852 NaN 1.9514538 -0.1934606 0.4980571   
## mf\_pred[34] mf\_pred[35] mf\_pred[36] mf\_pred[37] mf\_pred[38]   
## NaN NaN NaN NaN NaN   
## mf\_pred[39] mf\_pred[3] mf\_pred[40] mf\_pred[41] mf\_pred[42]   
## NaN 1.2341002 0.2707852 0.3662051 0.5401220   
## mf\_pred[43] mf\_pred[44] mf\_pred[45] mf\_pred[46] mf\_pred[47]   
## NaN 2.0881187 1.8380304 NaN 0.1724650   
## mf\_pred[48] mf\_pred[49] mf\_pred[4] mf\_pred[50] mf\_pred[51]   
## -0.1519945 NaN -2.5436537 0.2707852 0.4061458   
## mf\_pred[52] mf\_pred[53] mf\_pred[5] mf\_pred[6] mf\_pred[7]   
## 2.0226631 0.1341735 -1.0971250 0.8362895 -0.2182271   
## mf\_pred[8] mf\_pred[9] mf\_prod[10] mf\_prod[1] mf\_prod[2]   
## -0.1446816 2.3472610 1.3012732 -1.4747993 -0.5759622   
## mf\_prod[3] mf\_prod[4] mf\_prod[5] mf\_prod[6] mf\_prod[7]   
## 2.3603474 1.4472923 1.2249298 1.2410568 1.2387534   
## mf\_prod[8] mf\_prod[9] mu\_Intercept mu\_beta\_date mu\_beta\_whp   
## 1.2870500 1.2901733 -5.2589461 3.2815507 0.7692494   
## power[50] power[51] power[52] power[53] total\_power   
## 0.1922834 0.5150382 2.8401116 -0.1906677 1.4427615

gelman.diag(out[,c(paste0('mf\_pred[', 2:5, ']'), 'mu\_Intercept', 'total\_power')])[[1]] %>% as.data.frame() %>% round(2)

## Point est. Upper C.I.  
## mf\_pred[2] 1.26 1.86  
## mf\_pred[3] 1.02 1.09  
## mf\_pred[4] 1.00 1.01  
## mf\_pred[5] 1.01 1.01  
## mu\_Intercept 1.00 1.00  
## total\_power 1.01 1.03

raftery.diag(out[,c(paste0('mf\_pred[', 2:5, ']'), 'mu\_Intercept', 'total\_power')])

## [[1]]  
##   
## Quantile (q) = 0.025  
## Accuracy (r) = +/- 0.005  
## Probability (s) = 0.95   
##   
## You need a sample size of at least 3746 with these values of q, r and s  
##   
## [[2]]  
##   
## Quantile (q) = 0.025  
## Accuracy (r) = +/- 0.005  
## Probability (s) = 0.95   
##   
## You need a sample size of at least 3746 with these values of q, r and s