AnalysisAndPlots

Naitong Chen, Shirley Cui, Shannon Edie

3/10/2021

knitr::opts\_chunk$set(echo = TRUE)

# Read data

chardonnay2019 = read.csv("TanjaVoegel\_Chardonnay\_2019\_data.csv")  
chardonnay2020 = read.csv("TanjaVoegel\_Chardonnay\_2020\_data.csv")  
merlot2019 = read.csv("TanjaVoegel\_Merlot\_2019\_data.csv")  
merlot2020 = read.csv("TanjaVoegel\_Merlot\_2020\_data.csv")  
addition2021 = read.csv("data addition March 8 2021.csv")  
  
# removing extra columns  
merlot2020 = merlot2020[-(61:65),]  
merlot2020 = merlot2020[,-(19:20)]  
merlot2019 = merlot2019[-(61:65),]  
merlot2019 = merlot2019[,-18]  
  
# Make sure the merlot column names are consistent  
colnames(merlot2020)[colnames(merlot2020)=="spad"] = "SPAD"  
  
# encode row, treatment, block, and subsample as factor  
chardonnay2019$row = as.factor(chardonnay2019$row)  
chardonnay2019$treatment = as.factor(chardonnay2019$treatment)  
chardonnay2019$block = as.factor(chardonnay2019$block)  
chardonnay2019$subsample = as.factor(chardonnay2019$subsample)  
  
chardonnay2020$row = as.factor(chardonnay2020$row)  
chardonnay2020$treatment = as.factor(chardonnay2020$treatment)  
chardonnay2020$block = as.factor(chardonnay2020$block)  
chardonnay2020$subsample = as.factor(chardonnay2020$subsample)  
  
merlot2019$row = as.factor(merlot2019$row)  
merlot2019$treatment = as.factor(merlot2019$treatment)  
merlot2019$block = as.factor(merlot2019$block)  
merlot2019$subsample = as.factor(merlot2019$subsample)  
  
  
merlot2020$row = as.factor(merlot2020$row)  
merlot2020$treatment = as.factor(merlot2020$treatment)  
merlot2020$block = as.factor(merlot2020$block)  
merlot2020$subsample = as.factor(merlot2020$subsample)  
  
library(plyr)  
# Add in the additional data  
names(merlot2020)[1] <- "Key"  
merlot2020 <- join(merlot2020, addition2021)

## Joining by: Key, block, treatment, row, subsample

# detach plyr as it creates conflict with tidyverse  
detach(package:plyr)

# Load Packages

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.0.6 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## Warning: package 'tidyr' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(ggpubr)   
library(corrplot)

## corrplot 0.84 loaded

library(icenReg) # for survival analysis

## Loading required package: survival

## Loading required package: Rcpp

## Loading required package: coda

library(knitr)  
library(lawstat) # for the levene test  
library(mice)

##   
## Attaching package: 'mice'

## The following object is masked from 'package:stats':  
##   
## filter

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(miceadds)

## Warning: package 'miceadds' was built under R version 4.0.4

## \* miceadds 3.11-6 (2021-01-21 11:48:47)

library(MKinfer)

## Warning: package 'MKinfer' was built under R version 4.0.4

library(rstatix) # for the repeated-measures ANOVA

## Warning: package 'rstatix' was built under R version 4.0.4

##   
## Attaching package: 'rstatix'

## The following object is masked from 'package:stats':  
##   
## filter

library(ARTool) # for aligned rank transform

## Registered S3 methods overwritten by 'lme4':  
## method from  
## cooks.distance.influence.merMod car   
## influence.merMod car   
## dfbeta.influence.merMod car   
## dfbetas.influence.merMod car

library(maditr) # for dcast

##   
## To select rows from data: take\_if(mtcars, am==0)

##   
## Attaching package: 'maditr'

## The following objects are masked from 'package:dplyr':  
##   
## between, coalesce, first, last

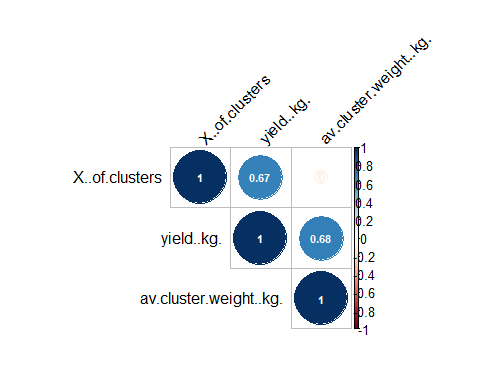
## The following object is masked from 'package:purrr':  
##   
## transpose

library(pwr2)

# EDA: correlation

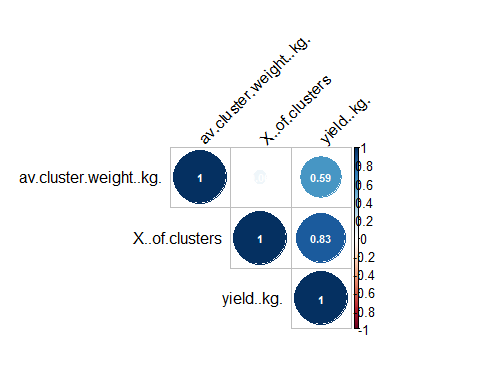
## chardonnay 2019

chardonnay2019 %>%   
 select(-1:-5) %>%   
 cor() %>%   
 corrplot(type = "upper", order = "hclust", tl.col = "black", tl.srt = 45, addCoef.col="white", number.cex=0.7)



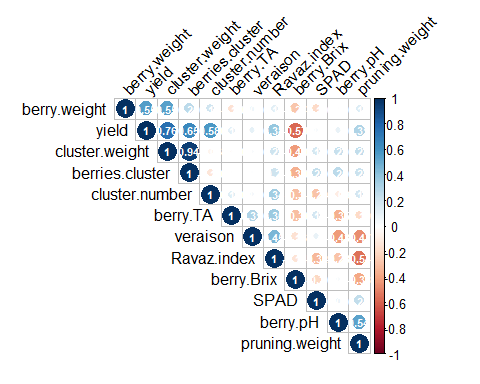
## chardonnay 2020

chardonnay2020 %>%   
 select(-1:-5) %>%   
 cor() %>%   
 corrplot(type = "upper", order = "hclust", tl.col = "black", tl.srt = 45, addCoef.col="white", number.cex=0.7)



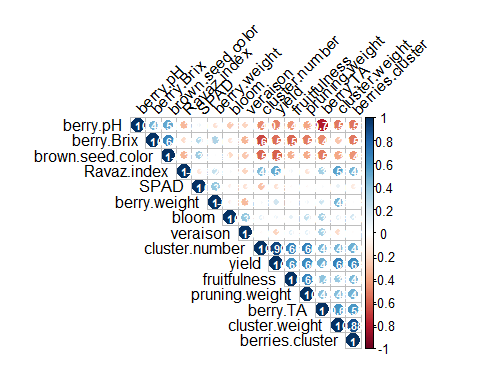
## merlot 2019

merlot2019.resp <- select(merlot2019,(-1:-5))  
merlot2019.resp %>%  
 drop\_na() %>%   
 cor() %>%  
 corrplot(type = "upper", order = "hclust", tl.col = "black", tl.srt = 45, addCoef.col="white", number.cex=0.7)



## merlot 2020

merlot2020.resp <- select(merlot2020,(-1:-5))  
merlot2020.resp %>%  
 drop\_na() %>%   
 cor() %>%  
 corrplot(type = "upper", order = "hclust", tl.col = "black", tl.srt = 45, addCoef.col="white", number.cex=0.7)

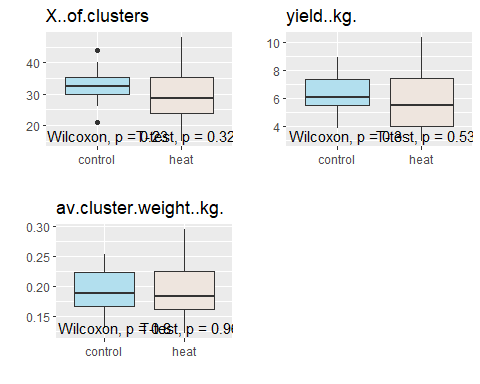


# EDA: Side by side box plots (ignoring blocks)

# helper  
plot\_resp = function(data, resp){  
 ggplot(data, aes(x=treatment, y = .data[[resp]], fill=treatment))+  
 scale\_fill\_manual(values=c("lightblue2", "seashell2"))+  
 ylab(" ") +  
 xlab(" ") +  
 labs(title = resp) +  
 theme(legend.position = "none")+  
 geom\_boxplot()+  
 stat\_compare\_means(method = "t.test", label.x = 1.7,  
 label.y = min(data[[resp]])) +  
 stat\_compare\_means(method = "wilcox.test", label.x = 0.7,  
 label.y = min(data[[resp]]))  
   
}

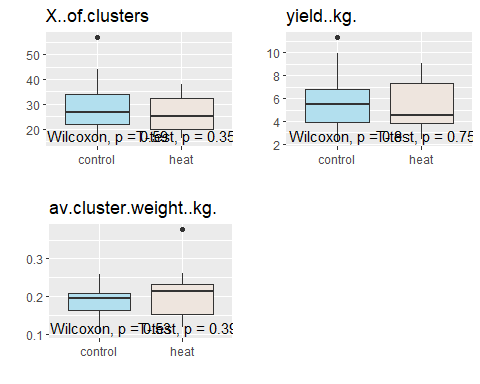
## chardonnay 2019

char2019.resp <- select(chardonnay2019,(-1:-5))  
plot\_list1= lapply(colnames(char2019.resp), plot\_resp, data = chardonnay2019)  
ggarrange(plotlist = plot\_list1)



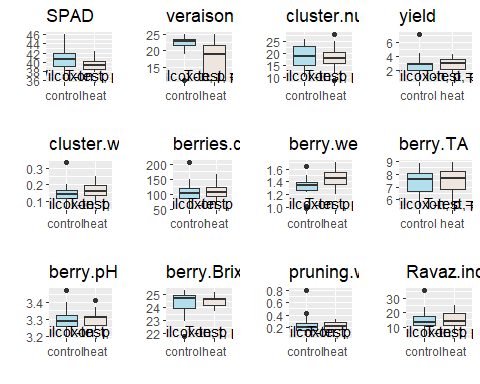
## chardonnay 2020

char2020.resp <- select(chardonnay2020, (-1:-5))  
plot\_list2=lapply(colnames(char2020.resp),plot\_resp, data=chardonnay2020)  
ggarrange(plotlist = plot\_list2)



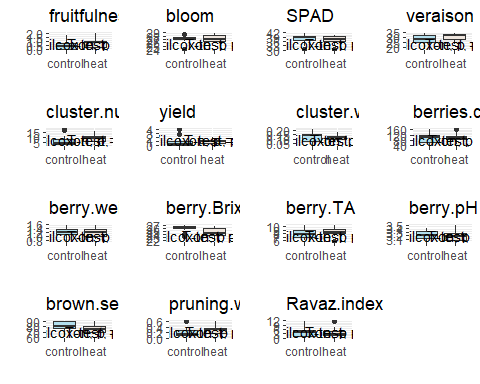
## merlot 2019

plot\_list3=lapply(colnames(merlot2019[-c(1:5)]), plot\_resp, data=drop\_na(merlot2019))  
ggarrange(plotlist = plot\_list3)



## merlot 2020

plot\_list4=lapply(colnames(merlot2020[-c(1:5)]), plot\_resp, data=drop\_na(merlot2020))  
ggarrange(plotlist = plot\_list4, heights = 2)

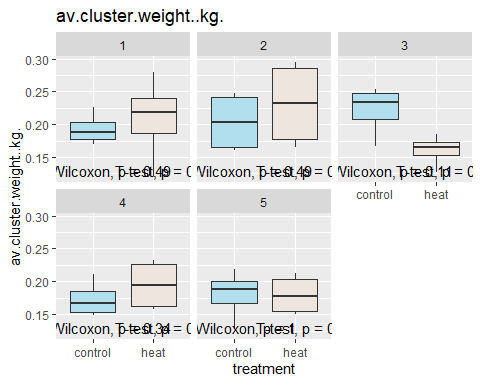
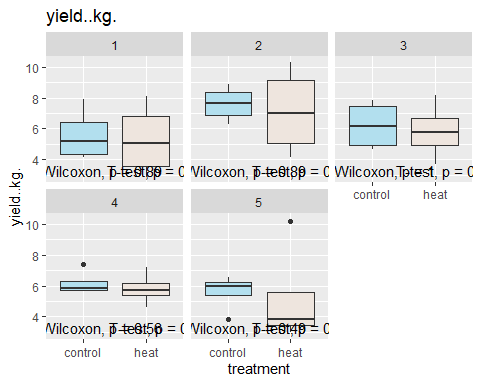
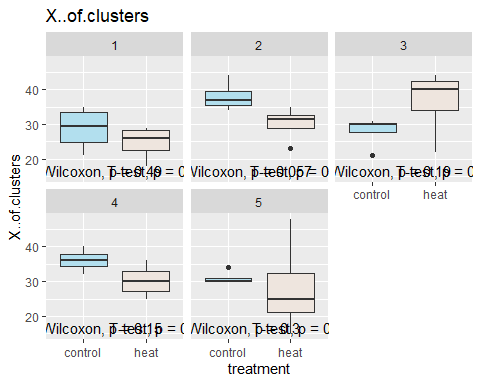


# EDA: Side by side box plots (treatment - blocks)

plot.trt.blk=function(data, resp)  
data %>%   
 ggplot(aes(x=treatment, y = .data[[resp]], fill=treatment))+  
 scale\_fill\_manual(values=c("lightblue2", "seashell2"))+  
 geom\_boxplot()+  
 labs(title = resp) +  
 theme(legend.position = "none")+  
 ylab(resp)+  
 facet\_wrap(~block)+  
 stat\_compare\_means(method = "wilcox.test", label.x = 0.7,  
 label.y = min(data[[resp]]))+  
 stat\_compare\_means(method = "t.test", label.x = 1.7,  
 label.y = min(data[[resp]]))

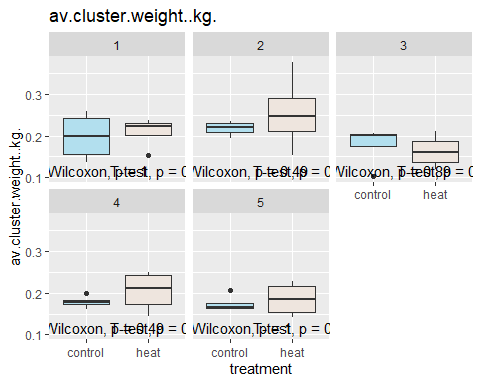
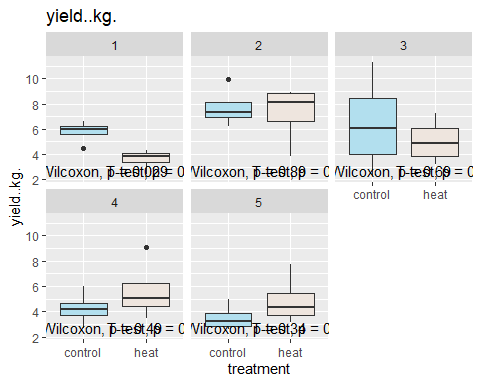
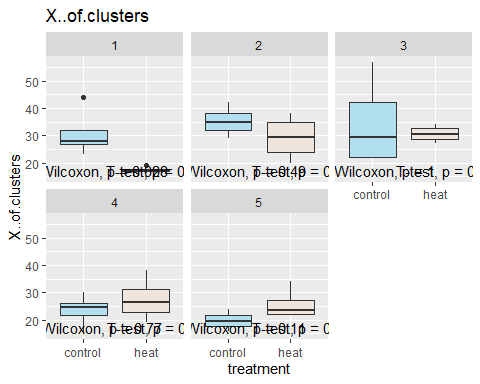
## chardonnay 2019

for (i in colnames(chardonnay2019[-1:-5])){  
 print(plot.trt.blk(chardonnay2019, i))  
}



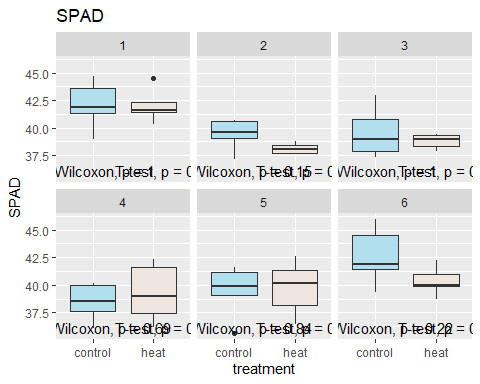
## chardonnay 2020

for (i in colnames(chardonnay2020[-1:-5])){  
 print(plot.trt.blk(chardonnay2020, i))  
}



## merlot 2019

for (i in colnames(merlot2019[-1:-5])){  
 print(plot.trt.blk(merlot2019, i))  
}



## Warning: Removed 26 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 26 rows containing non-finite values (stat\_compare\_means).

## Warning: Computation failed in `stat\_compare\_means()`:  
## argument "x" is missing, with no default

## Warning: Removed 26 rows containing non-finite values (stat\_compare\_means).

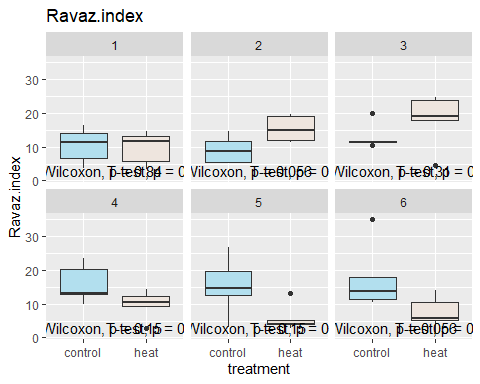
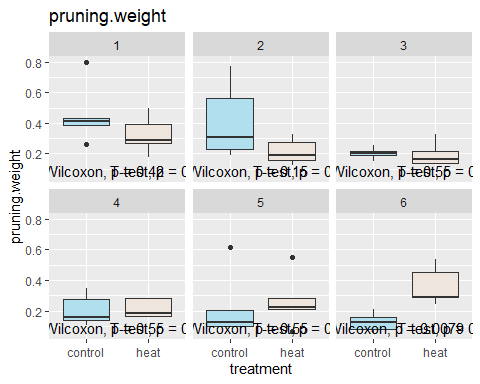
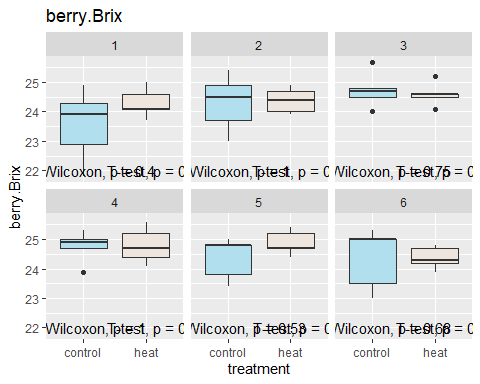
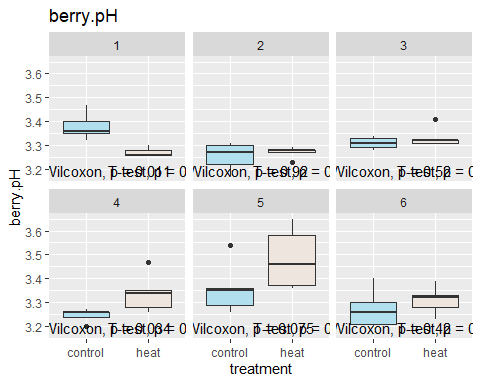
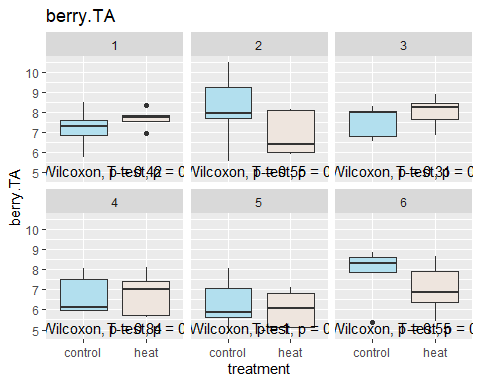
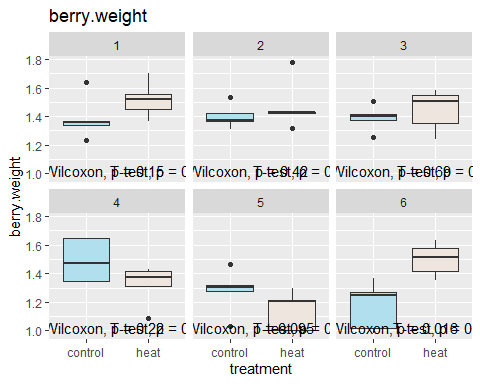
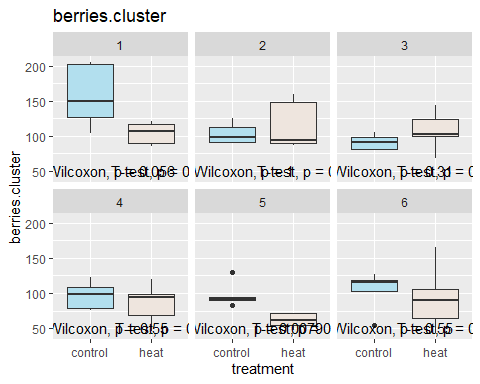
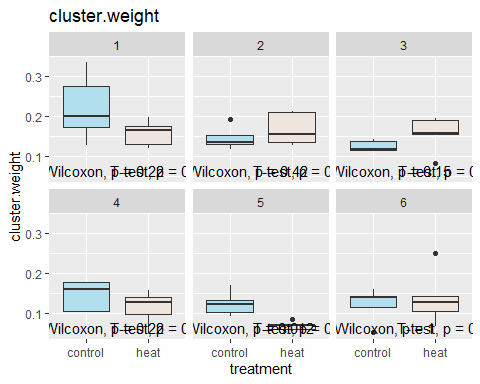
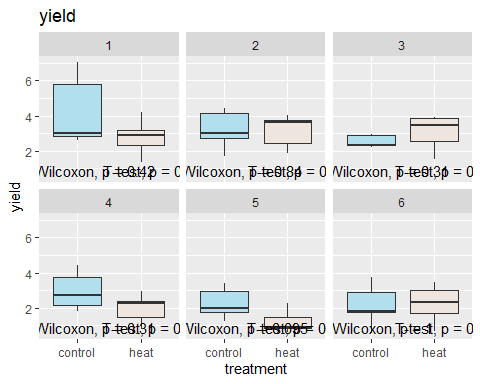
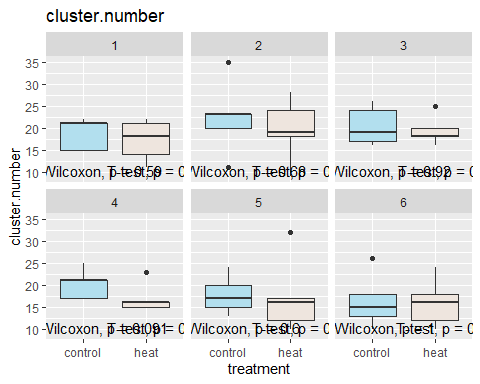
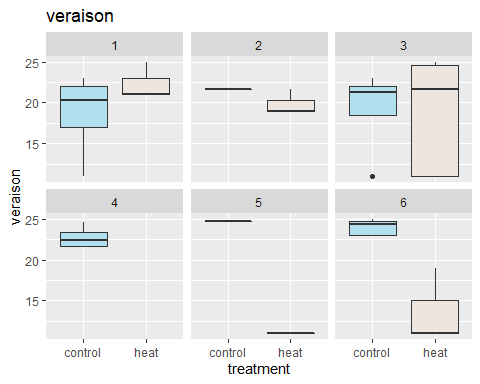
## Warning: Computation failed in `stat\_compare\_means()`:  
## not enough 'x' observations

## Warning: Computation failed in `stat\_compare\_means()`:  
## argument "x" is missing, with no default

## Warning: Computation failed in `stat\_compare\_means()`:  
## not enough 'x' observations

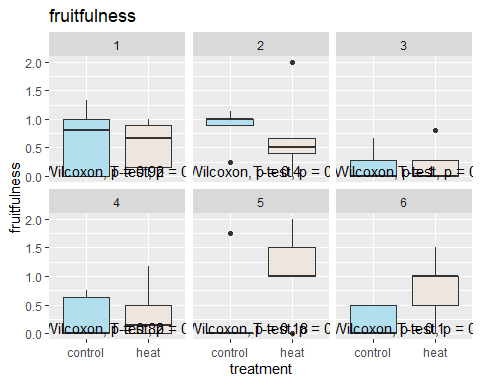
## Warning: Removed 5 rows containing missing values (geom\_text).

## Warning: Removed 3 rows containing missing values (geom\_text).



## merlot 2020

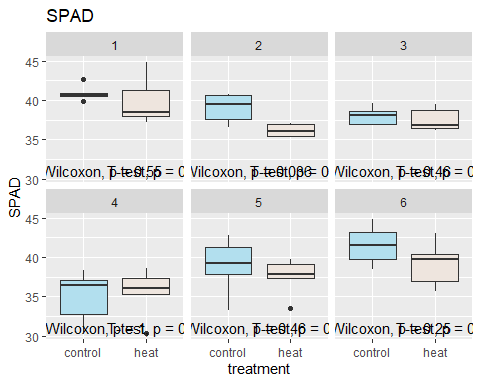
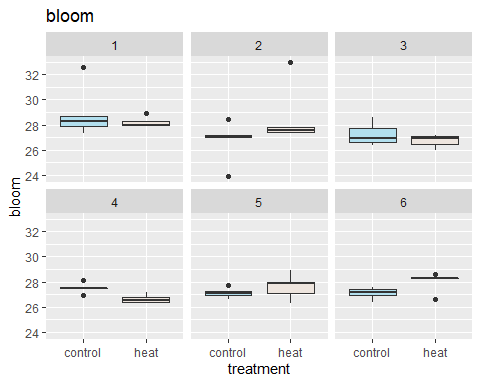
for (i in colnames(merlot2020[-1:-5])){  
 print(plot.trt.blk(merlot2020, i))  
}



## Warning: Removed 7 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).

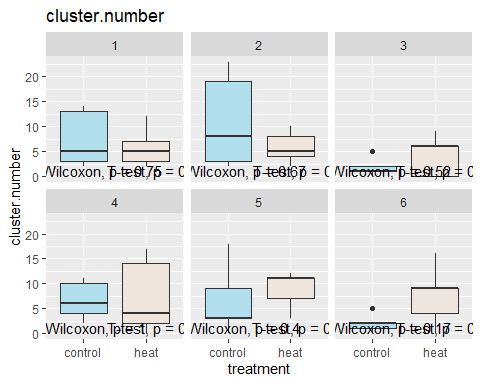
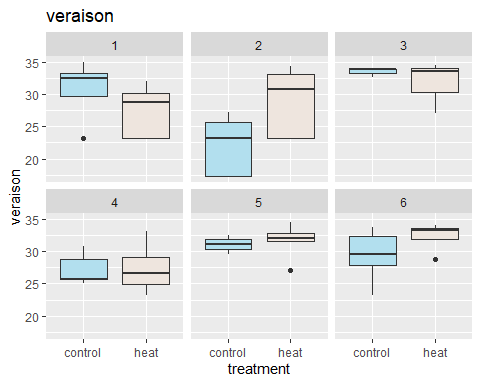
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 8 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).

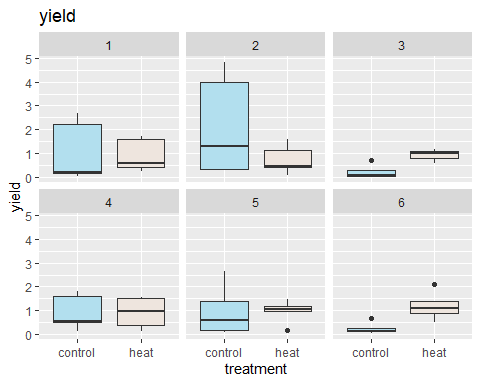
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 7 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).

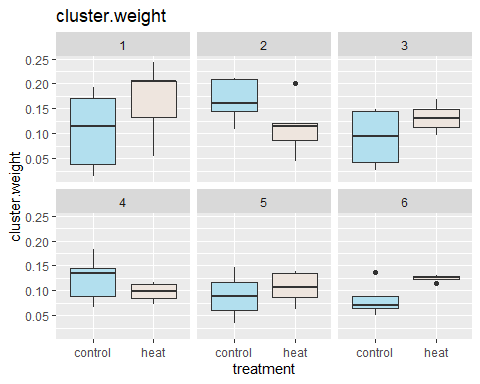
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 7 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).

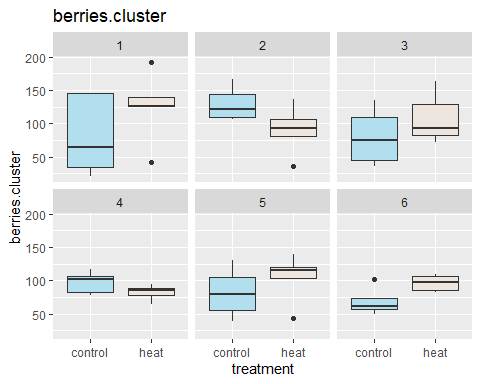
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 7 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).

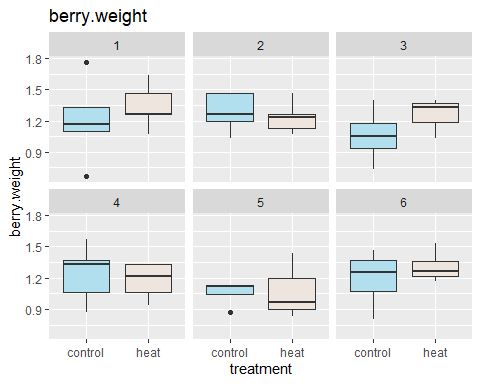
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 7 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 7 rows containing non-finite values (stat\_compare\_means).

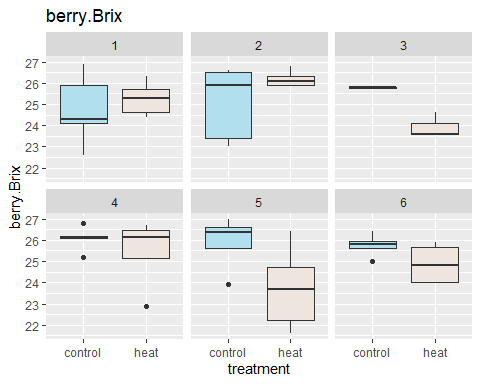
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 8 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).

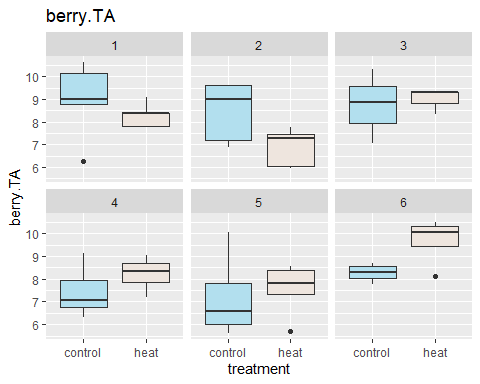
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 8 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).

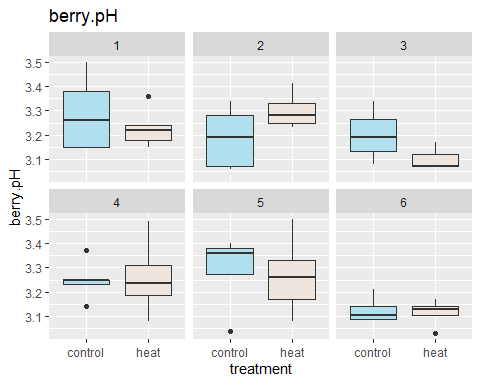
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 8 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).

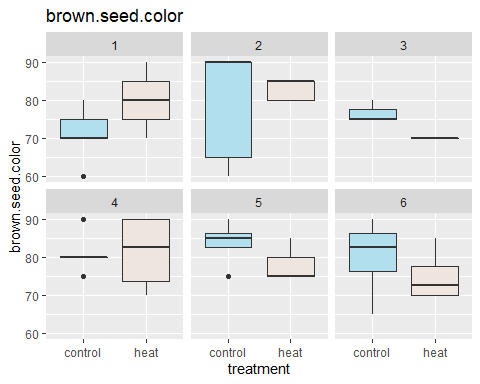
## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 8 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).  
  
## Warning: Removed 8 rows containing non-finite values (stat\_compare\_means).

## Warning: Removed 6 rows containing missing values (geom\_text).  
  
## Warning: Removed 6 rows containing missing values (geom\_text).



## Warning: Removed 16 rows containing non-finite values (stat\_boxplot).

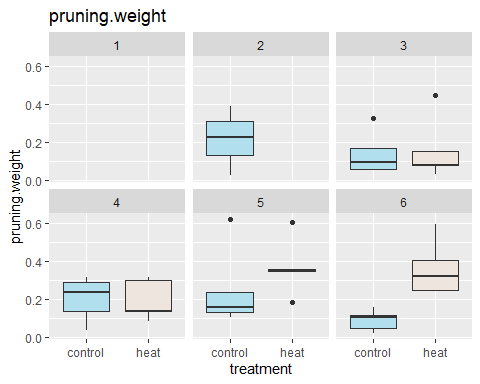
## Warning: Removed 16 rows containing non-finite values (stat\_compare\_means).

## Warning: Computation failed in `stat\_compare\_means()`:  
## argument "x" is missing, with no default

## Warning: Removed 16 rows containing non-finite values (stat\_compare\_means).

## Warning: Computation failed in `stat\_compare\_means()`:  
## argument "x" is missing, with no default

## Warning: Removed 4 rows containing missing values (geom\_text).  
  
## Warning: Removed 4 rows containing missing values (geom\_text).



## Warning: Removed 23 rows containing non-finite values (stat\_boxplot).

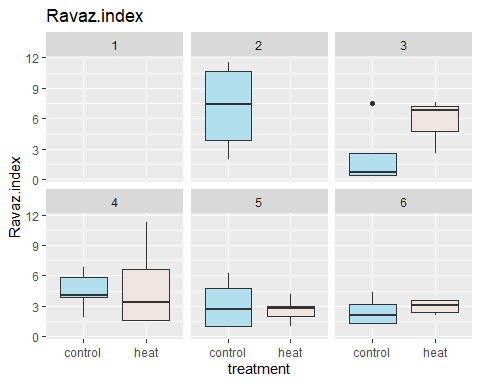
## Warning: Removed 23 rows containing non-finite values (stat\_compare\_means).

## Warning: Computation failed in `stat\_compare\_means()`:  
## argument "x" is missing, with no default

## Warning: Removed 23 rows containing non-finite values (stat\_compare\_means).

## Warning: Computation failed in `stat\_compare\_means()`:  
## argument "x" is missing, with no default

## Warning: Removed 4 rows containing missing values (geom\_text).  
  
## Warning: Removed 4 rows containing missing values (geom\_text).



# Censored data analysis

# Missing data rates

## merlot 2019

print("veraison")

## [1] "veraison"

paste0(round(mean(is.na(merlot2019$veraison)), 3)\*100, "%")

## [1] "43.3%"

## merlot 2020

print("bloom")

## [1] "bloom"

paste0(round(mean(is.na(merlot2020$bloom)), 3)\*100, "%")

## [1] "11.7%"

print("veraison")

## [1] "veraison"

paste0(round(mean(is.na(merlot2020$veraison)), 3)\*100, "%")

## [1] "13.3%"

print("yield, cluster weight, berries.cluster, berry.weight")

## [1] "yield, cluster weight, berries.cluster, berry.weight"

paste0(round(mean(is.na(merlot2020$yield)), 3)\*100, "%")

## [1] "11.7%"

print("berry Brix, berry pH, berry TA, brown.seed.color")

## [1] "berry Brix, berry pH, berry TA, brown.seed.color"

paste0(round(mean(is.na(merlot2020$brown.seed.color)), 3)\*100, "%")

## [1] "13.3%"

print("pruning.weight")

## [1] "pruning.weight"

paste0(round(mean(is.na(merlot2020$pruning.weight)), 3)\*100, "%")

## [1] "26.7%"

print("Ravaz.index")

## [1] "Ravaz.index"

paste0(round(mean(is.na(merlot2020$Ravaz.index)), 3)\*100, "%")

## [1] "38.3%"

# Complete case analysis: Randomized Block Design

## create a helper function of aov for complete randomized block design  
comp.rand.blk<- function(data, resp) {  
 data %>%   
 select(block, treatment, resp) %>%   
 group\_by(treatment, block) %>%   
 mutate(mresp = mean(.data[[resp]], na.rm=TRUE)) %>%   
 select(treatment, block, mresp) %>%   
 distinct() -> newdata  
 print(resp)  
 return(summary(aov(mresp~treatment+block,data=newdata)))  
}

## chardonnay 2019

resps = colnames(chardonnay2019)[6:8]  
char.2019.cc = array(data=rep(0, length(resps) \* 2), dim=c(length(resps), 2))  
dimnames(char.2019.cc)[[2]] <- c("treatment", "block")  
dimnames(char.2019.cc)[[1]] <- resps  
  
for (i in 1:length(resps)) {  
 res = comp.rand.blk(chardonnay2019, resps[i])  
 char.2019.cc[i,] = res[[1]][["Pr(>F)"]][1:2]  
 print(res)  
}

## Note: Using an external vector in selections is ambiguous.  
## i Use `all\_of(resp)` instead of `resp` to silence this message.  
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.  
## This message is displayed once per session.

## [1] "X..of.clusters"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 13.81 13.81 0.685 0.454  
## block 4 71.29 17.82 0.884 0.546  
## Residuals 4 80.66 20.17   
## [1] "yield..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.363 0.3629 218.8 0.000122 \*\*\*  
## block 4 5.111 1.2777 770.2 5.04e-06 \*\*\*  
## Residuals 4 0.007 0.0017   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## [1] "av.cluster.weight..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.0000014 0.0000014 0.002 0.965  
## block 4 0.0017912 0.0004478 0.692 0.635  
## Residuals 4 0.0025897 0.0006474

print(char.2019.cc)

## treatment block  
## X..of.clusters 0.4544990572 5.462147e-01  
## yield..kg. 0.0001216116 5.039254e-06  
## av.cluster.weight..kg. 0.9648615366 6.351722e-01

## chardonnay 2020

resps = colnames(chardonnay2020)[6:8]  
char.2020.cc = array(data=rep(0, length(resps) \* 2), dim=c(length(resps), 2))  
dimnames(char.2020.cc)[[2]] <- c("treatment", "block")  
dimnames(char.2020.cc)[[1]] <- resps  
for (i in 1:length(resps)) {  
 res = comp.rand.blk(chardonnay2020, resps[i])  
 char.2020.cc[i,] = res[[1]][["Pr(>F)"]][1:2]  
 print(res)  
}

## [1] "X..of.clusters"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 18.23 18.23 0.584 0.487  
## block 4 172.65 43.16 1.382 0.381  
## Residuals 4 124.90 31.22   
## [1] "yield..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.135 0.135 0.104 0.763  
## block 4 13.285 3.321 2.561 0.192  
## Residuals 4 5.188 1.297   
## [1] "av.cluster.weight..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.000476 0.0004756 2.454 0.1923   
## block 4 0.005377 0.0013442 6.936 0.0436 \*  
## Residuals 4 0.000775 0.0001938   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

print(char.2020.cc)

## treatment block  
## X..of.clusters 0.4874564 0.38067527  
## yield..kg. 0.7628207 0.19230671  
## av.cluster.weight..kg. 0.1922890 0.04363301

## merlot 2019

resps = colnames(merlot2019)[6:17]  
resps = resps[-2] # remove veraison  
mer.2019.cc = array(data=rep(0, length(resps) \* 2), dim=c(length(resps), 2))  
dimnames(mer.2019.cc)[[2]] <- c("treatment", "block")  
dimnames(mer.2019.cc)[[1]] <- resps  
for (i in 1:length(resps)) {  
 res = comp.rand.blk(merlot2019, resps[i])  
 mer.2019.cc[i,] = res[[1]][["Pr(>F)"]][1:2]  
 print(res)  
}

## [1] "SPAD"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.963 0.963 1.519 0.2726   
## block 5 19.985 3.997 6.302 0.0323 \*  
## Residuals 5 3.171 0.634   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## [1] "cluster.number"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 7.363 7.363 10.180 0.0242 \*  
## block 5 29.110 5.822 8.049 0.0195 \*  
## Residuals 5 3.617 0.723   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## [1] "yield"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.754 0.7538 2.444 0.179  
## block 5 4.205 0.8411 2.727 0.148  
## Residuals 5 1.542 0.3084   
## [1] "cluster.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.000555 0.0005550 0.637 0.461  
## block 5 0.009757 0.0019515 2.238 0.199  
## Residuals 5 0.004359 0.0008718   
## [1] "berries.cluster"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 561.8 561.8 1.486 0.277  
## block 5 3096.0 619.2 1.637 0.301  
## Residuals 5 1890.8 378.2   
## [1] "berry.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.00677 0.006769 0.417 0.547  
## block 5 0.08342 0.016684 1.029 0.488  
## Residuals 5 0.08110 0.016219   
## [1] "berry.TA"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.124 0.1242 0.497 0.512  
## block 5 3.673 0.7346 2.939 0.131  
## Residuals 5 1.250 0.2499   
## [1] "berry.pH"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.002883 0.002883 0.894 0.388  
## block 5 0.030639 0.006128 1.901 0.249  
## Residuals 5 0.016119 0.003224   
## [1] "berry.Brix"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.1323 0.13230 2.276 0.192  
## block 5 0.9502 0.19003 3.269 0.110  
## Residuals 5 0.2907 0.05814   
## [1] "pruning.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.00026 0.000261 0.023 0.886  
## block 5 0.05322 0.010644 0.929 0.531  
## Residuals 5 0.05728 0.011457   
## [1] "Ravaz.index"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 19.60 19.602 0.752 0.426  
## block 5 37.19 7.437 0.285 0.903  
## Residuals 5 130.36 26.073

print(mer.2019.cc)

## treatment block  
## SPAD 0.27256740 0.03232039  
## cluster.number 0.02424957 0.01950095  
## yield 0.17873634 0.14750315  
## cluster.weight 0.46111305 0.19859131  
## berries.cluster 0.27726898 0.30078166  
## berry.weight 0.54674921 0.48800364  
## berry.TA 0.51226115 0.13087290  
## berry.pH 0.38774128 0.24893289  
## berry.Brix 0.19180051 0.10979508  
## pruning.weight 0.88585472 0.53118986  
## Ravaz.index 0.42554357 0.90257628

## merlot 2020

resps = colnames(merlot2020)[6:20]  
resps = resps[-c(2,4)] # remove veraison and bloom  
mer.2020.cc = array(data=rep(0, length(resps) \* 2), dim=c(length(resps), 2))  
dimnames(mer.2020.cc)[[2]] <- c("treatment", "block")  
dimnames(mer.2020.cc)[[1]] <- resps  
for (i in 1:length(resps)) {  
 res = comp.rand.blk(merlot2020, resps[i])  
 mer.2020.cc[i,] = res[[1]][["Pr(>F)"]][1:2]  
 print(res)  
}

## [1] "fruitfulness"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.1284 0.12842 1.830 0.234  
## block 5 0.5137 0.10274 1.464 0.343  
## Residuals 5 0.3508 0.07017   
## [1] "SPAD"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 4.79 4.788 6.32 0.0536 .  
## block 5 38.15 7.630 10.07 0.0121 \*  
## Residuals 5 3.79 0.758   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## [1] "cluster.number"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 1.47 1.470 0.212 0.664  
## block 5 40.15 8.030 1.159 0.438  
## Residuals 5 34.63 6.926   
## [1] "yield"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.0002 0.0002 0.001 0.982  
## block 5 0.8804 0.1761 0.532 0.747  
## Residuals 5 1.6552 0.3310   
## [1] "cluster.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.000549 0.0005492 0.543 0.494  
## block 5 0.003136 0.0006271 0.620 0.694  
## Residuals 5 0.005056 0.0010112   
## [1] "berries.cluster"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 404.2 404.2 0.829 0.404  
## block 5 954.5 190.9 0.391 0.837  
## Residuals 5 2438.4 487.7   
## [1] "berry.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.00948 0.009477 1.621 0.259  
## block 5 0.06455 0.012911 2.208 0.203  
## Residuals 5 0.02924 0.005847   
## [1] "berry.Brix"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 1.250 1.2502 1.516 0.273  
## block 5 1.688 0.3375 0.409 0.825  
## Residuals 5 4.125 0.8249   
## [1] "berry.TA"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.027 0.0268 0.047 0.836  
## block 5 4.564 0.9129 1.609 0.307  
## Residuals 5 2.836 0.5672   
## [1] "berry.pH"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.00039 0.000391 0.151 0.714  
## block 5 0.04183 0.008366 3.229 0.112  
## Residuals 5 0.01296 0.002591   
## [1] "brown.seed.color"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 1.45 1.447 0.074 0.797  
## block 5 110.03 22.005 1.119 0.452  
## Residuals 5 98.34 19.668   
## [1] "pruning.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.01818 0.018176 2.249 0.231  
## block 4 0.02869 0.007173 0.887 0.562  
## Residuals 3 0.02425 0.008083   
## 3 observations deleted due to missingness  
## [1] "Ravaz.index"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.044 0.044 0.030 0.873  
## block 4 18.355 4.589 3.163 0.186  
## Residuals 3 4.353 1.451   
## 3 observations deleted due to missingness

print(mer.2020.cc)

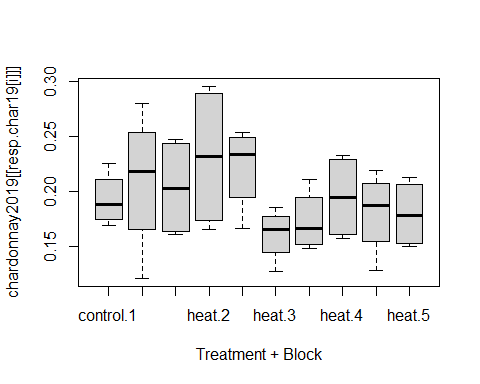
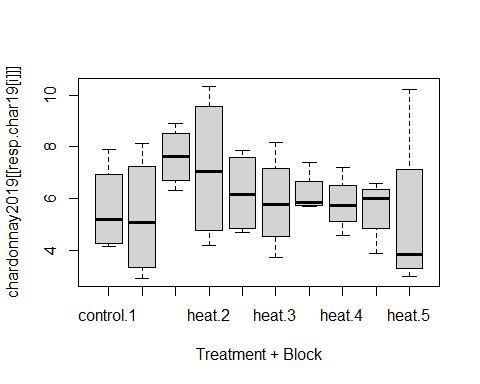
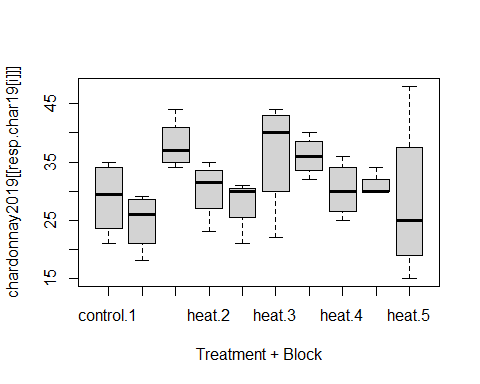
## treatment block  
## fruitfulness 0.23406936 0.34291951  
## SPAD 0.05356308 0.01205399  
## cluster.number 0.66435840 0.43751315  
## yield 0.98222550 0.74744401  
## cluster.weight 0.49427292 0.69357734  
## berries.cluster 0.40434223 0.83678901  
## berry.weight 0.25896911 0.20253511  
## berry.Brix 0.27303402 0.82546160  
## berry.TA 0.83640427 0.30714372  
## berry.pH 0.71365438 0.11207378  
## brown.seed.color 0.79706263 0.45246258  
## pruning.weight 0.23068606 0.56200239  
## Ravaz.index 0.87300805 0.18568149

# Multiple Imputation

# Two-way ANOVA assumption checks

## chardonnay 2019 equal variance

resp.char19 <- colnames(chardonnay2019)[-c(1:5)]  
homosc.table.char19 <- array(data=NA, dim=c(length(resp.char19), 2))  
dimnames(homosc.table.char19)[[2]] <- c("Bartlett", "Levene")  
dimnames(homosc.table.char19)[[1]] <- resp.char19  
  
for (i in 1:length(resp.char19)) {  
  
 # the lapply function simply extracts the p-value from each outputted run of the MICE data  
 bart.p <- with(chardonnay2019,   
 bartlett.test(eval(parse(text=resp.char19[i])) ~ paste(treatment, block)))$p.value  
 lev.p <- with(chardonnay2019,   
 levene.test(eval(parse(text=resp.char19[i])), paste(treatment, block)))$p.value  
 homosc.table.char19[i,1] <- bart.p  
 homosc.table.char19[i,2] <- lev.p  
}  
  
for (i in 1:length(resp.char19)) {  
 boxplot(chardonnay2019[[resp.char19[i]]] ~ chardonnay2019$treatment + chardonnay2019$block,  
 xlab="Treatment + Block")  
}

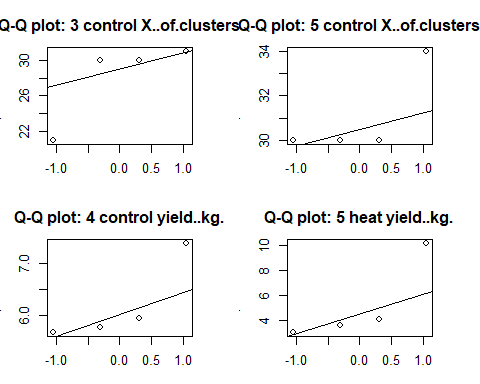


homosc.table.char19

## Bartlett Levene  
## X..of.clusters 0.1054019 0.4605184  
## yield..kg. 0.3694111 0.4071546  
## av.cluster.weight..kg. 0.6834928 0.1453085

## chardonnay 2019 normality

# Run shapiro-wilkes tests on each of the variables we will run ANOVAs on  
norm.tests.char19 <- array(NA, c(10, length(resp.char19)))  
colnames(norm.tests.char19) <- resp.char19  
rownames(norm.tests.char19) <- unique(paste(chardonnay2019$block, chardonnay2019$treatment))[order(unique(paste(chardonnay2019$block, chardonnay2019$treatment)))]  
  
# Run the shapiro wilkes test on each of the outcome variables in the chardonnay 2019 dataset  
for (i in 1:length(resp.char19)) {  
 shapiro.results <- with(chardonnay2019, {lapply(split(eval(parse(text=resp.char19[i])),   
 paste0(block, treatment)),   
 shapiro.test)})  
 # Save the shapiro wilkes tests for each of the populations (unique combination of block & treatment)  
 # in one large array  
 for (k in 1:10) {norm.tests.char19[k,i] <- shapiro.results[[k]]$p.value}  
}  
  
# Find the row and column names of the potentially non-normal data  
non.norm <- which(norm.tests.char19<0.05, arr.ind=T)  
non.norm[,2] <- colnames(norm.tests.char19)[non.norm[,2]]  
non.norm[,1] <- rownames(norm.tests.char19)[as.numeric(non.norm[,1])]  
# Plot the Q-Q plots of the potentially non-normal groups  
par(mfrow=c(2,2), mar=c(3,3,3,3))  
for (i in 1:dim(non.norm)[1]) {  
 qqnorm(chardonnay2019[which(paste(chardonnay2019$block, chardonnay2019$treatment)==non.norm[i,1]),non.norm[i,2]],  
 main=paste("Q-Q plot:", non.norm[i,1], non.norm[i,2]))  
 qqline(chardonnay2019[which(paste(chardonnay2019$block, chardonnay2019$treatment)==non.norm[i,1]),non.norm[i,2]])  
}

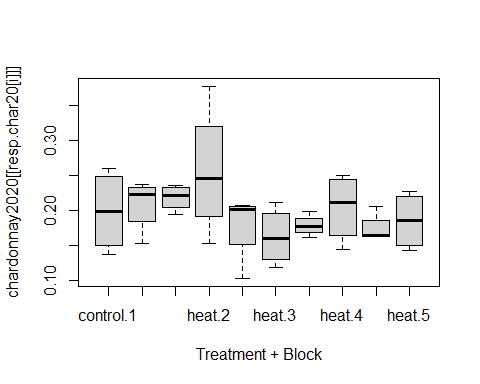
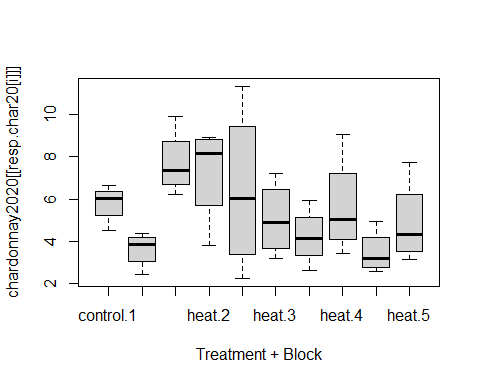
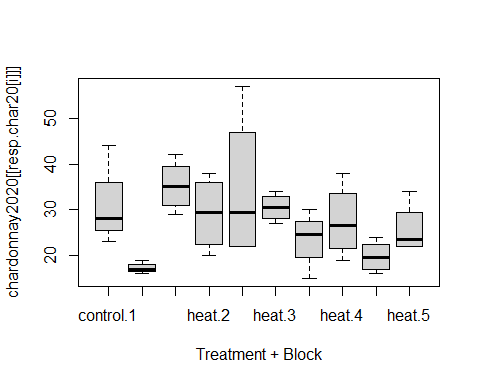


norm.tests.char19

## X..of.clusters yield..kg. av.cluster.weight..kg.  
## 1 control 0.609626805 0.43426525 0.73075416  
## 1 heat 0.456731068 0.65315877 0.78484769  
## 2 control 0.577354902 0.86702000 0.08826218  
## 2 heat 0.425536857 0.57021186 0.16907363  
## 3 control 0.017407956 0.21477290 0.31624836  
## 3 heat 0.176703042 0.97550895 0.64635802  
## 4 control 0.999855866 0.03949909 0.51402140  
## 4 heat 0.925444473 0.77208097 0.11800466  
## 5 control 0.001240726 0.21521157 0.73010290  
## 5 heat 0.516618270 0.03629821 0.28413899

## chardonnay 2020 equal variance

resp.char20 <- colnames(chardonnay2020)[-c(1:5)]  
homosc.table.char20 <- array(data=NA, dim=c(length(resp.char20), 2))  
dimnames(homosc.table.char20)[[2]] <- c("Bartlett", "Levene")  
dimnames(homosc.table.char20)[[1]] <- resp.char19  
  
for (i in 1:length(resp.char20)) {  
  
 # the lapply function simply extracts the p-value from each outputted run of the MICE data  
 bart.p <- with(chardonnay2020,   
 bartlett.test(eval(parse(text=resp.char20[i])) ~ paste(treatment, block)))$p.value  
 lev.p <- with(chardonnay2020,   
 levene.test(eval(parse(text=resp.char20[i])), paste(treatment, block)))$p.value  
 homosc.table.char20[i,1] <- bart.p  
 homosc.table.char20[i,2] <- lev.p  
}  
  
for (i in 1:length(resp.char20)) {  
 boxplot(chardonnay2020[[resp.char20[i]]] ~ chardonnay2020$treatment + chardonnay2020$block,  
 xlab="Treatment + Block")  
}



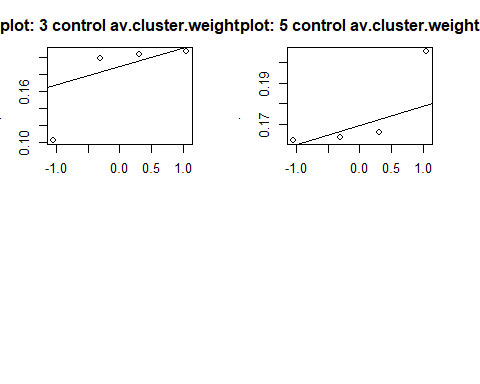
homosc.table.char20

## Bartlett Levene  
## X..of.clusters 0.02804167 0.1176924  
## yield..kg. 0.24393329 0.2651053  
## av.cluster.weight..kg. 0.14176257 0.1852972

## chardonnay 2020 normality

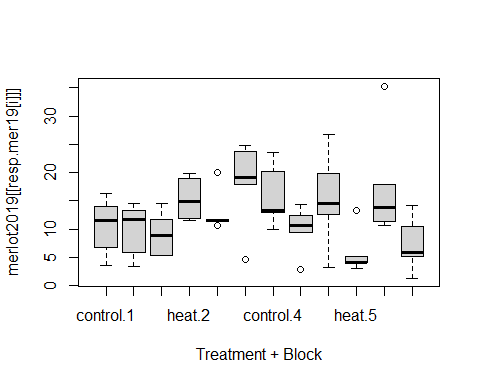
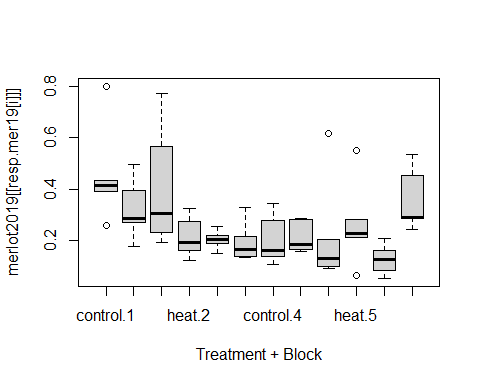
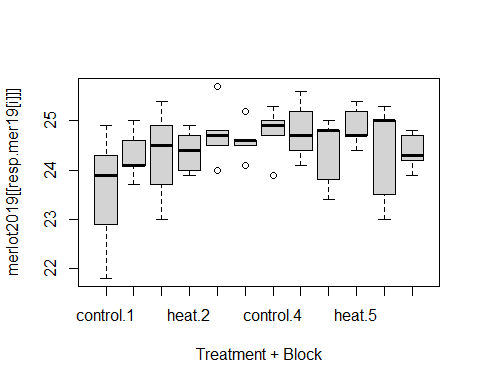
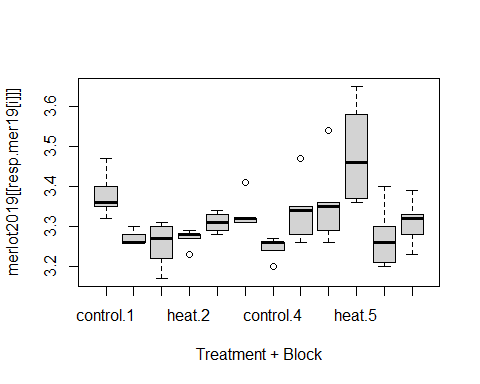
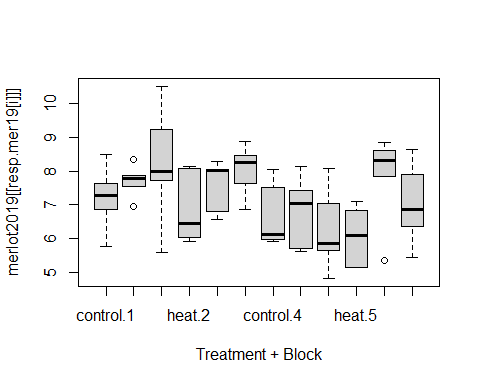
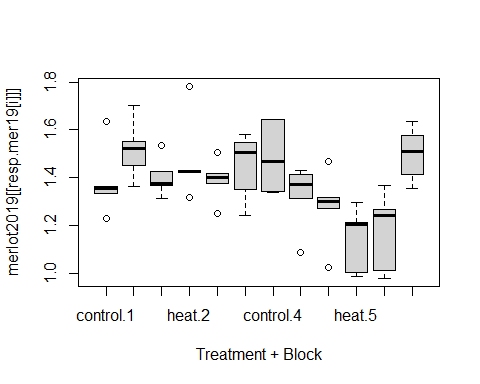
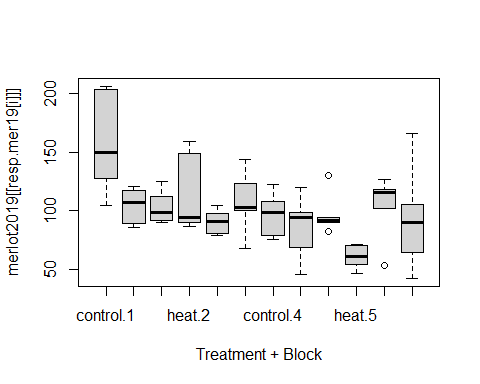
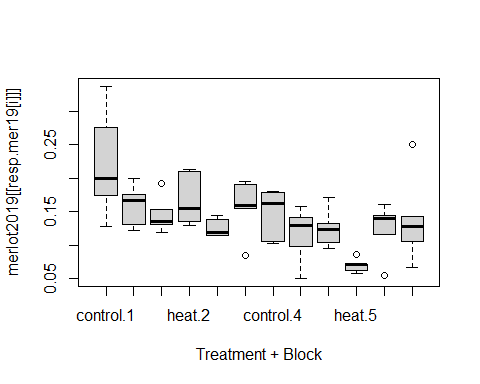
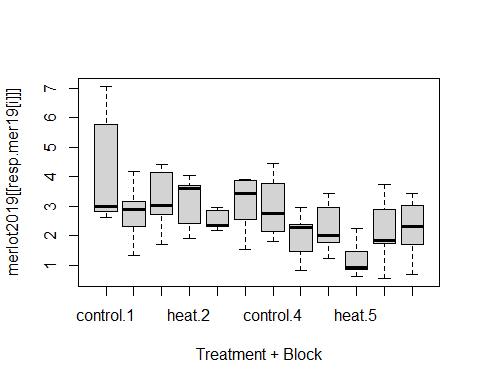
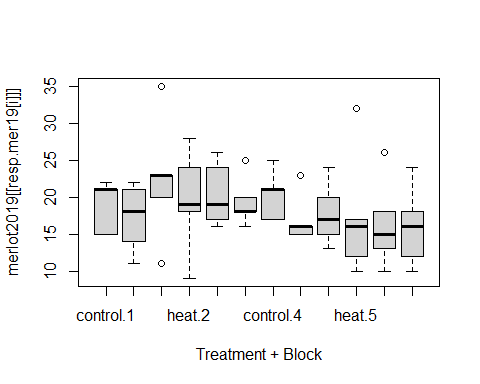
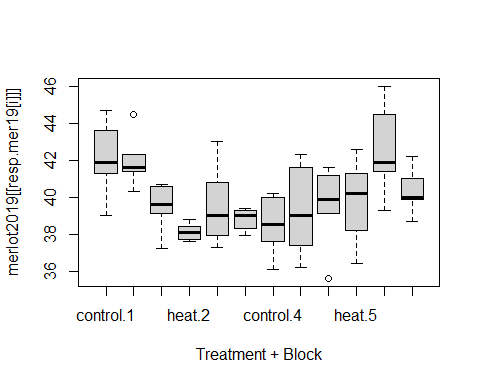
# Run shapiro-wilkes tests on each of the variables we will run ANOVAs on  
norm.tests.char20 <- array(NA, c(10, length(resp.char20)))  
colnames(norm.tests.char20) <- resp.char20  
rownames(norm.tests.char20) <- unique(paste(chardonnay2020$block, chardonnay2020$treatment))[order(unique(paste(chardonnay2020$block, chardonnay2020$treatment)))]  
  
# Run the shapiro wilkes test on each of the outcome variables in the chardonnay 2019 dataset  
for (i in 1:length(resp.char20)) {  
 shapiro.results <- with(chardonnay2020, {lapply(split(eval(parse(text=resp.char20[i])),   
 paste0(block, treatment)),   
 shapiro.test)})  
 # Save the shapiro wilkes tests for each of the populations (unique combination of block & treatment)  
 # in one large array  
 for (k in 1:10) {norm.tests.char20[k,i] <- shapiro.results[[k]]$p.value}  
}  
  
# Find the row and column names of the potentially non-normal data  
non.norm <- which(norm.tests.char20<0.05, arr.ind=T)  
non.norm[,2] <- colnames(norm.tests.char20)[non.norm[,2]]  
non.norm[,1] <- rownames(norm.tests.char20)[as.numeric(non.norm[,1])]  
# Plot the Q-Q plots of the potentially non-normal groups  
par(mfrow=c(2,2), mar=c(3,3,3,3))  
for (i in 1:dim(non.norm)[1]) {  
 qqnorm(chardonnay2020[which(paste(chardonnay2020$block, chardonnay2020$treatment)==non.norm[i,1]),non.norm[i,2]],  
 main=paste("Q-Q plot:", non.norm[i,1], non.norm[i,2]))  
 qqline(chardonnay2020[which(paste(chardonnay2020$block, chardonnay2020$treatment)==non.norm[i,1]),non.norm[i,2]])  
}  
  
norm.tests.char20

## X..of.clusters yield..kg. av.cluster.weight..kg.  
## 1 control 0.16963968 0.3468788 0.445168262  
## 1 heat 0.40638745 0.4227822 0.149799818  
## 2 control 0.97508238 0.5108555 0.756945912  
## 2 heat 0.70697709 0.1200417 0.880804876  
## 3 control 0.24066991 0.9105084 0.009340648  
## 3 heat 0.85522685 0.8383605 0.867564947  
## 4 control 0.64144078 0.8228552 0.677624254  
## 4 heat 0.89795340 0.4281771 0.549065700  
## 5 control 0.89407308 0.4567407 0.010700013  
## 5 heat 0.08648694 0.3988850 0.344138026



## merlot 2019 equal variance

# excluding veraison  
resp.mer19 <- colnames(merlot2019)[-c(1:5, 7)]  
  
# Create some tables to save the results of our statistical tests in  
homosc.table.mer19 <- array(data=NA, dim=c(length(resp.mer19), 2))  
dimnames(homosc.table.mer19)[[2]] <- c("Bartlett", "Levene")  
dimnames(homosc.table.mer19)[[1]] <- resp.mer19  
  
# Compare variances for each unique treatment-block group  
for (i in 1:length(resp.mer19)) {  
  
 # the lapply function simply extracts the p-value from each outputted run of the MICE data  
 bart.p <- (with(merlot2019,  
 bartlett.test(eval(parse(text=resp.mer19[i])) ~ paste(treatment, block)))$p.value)  
 lev.p <- (with(merlot2019,  
 levene.test(eval(parse(text=resp.mer19[i])), paste(treatment, block)))$p.value)  
 homosc.table.mer19[i,1] <- bart.p  
 homosc.table.mer19[i,2] <- lev.p  
}  
  
for (i in 1:length(resp.mer19)) {  
 boxplot(merlot2019[[resp.mer19[i]]] ~ merlot2019$treatment + merlot2019$block,  
 xlab="Treatment + Block")  
}

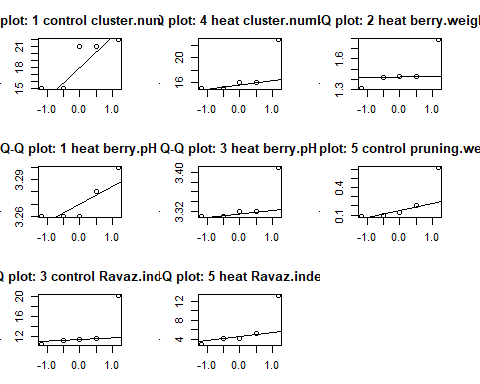


homosc.table.mer19

## Bartlett Levene  
## SPAD 0.13642878 0.37240716  
## cluster.number 0.47661999 0.88501921  
## yield 0.37826156 0.78129501  
## cluster.weight 0.02980026 0.33965014  
## berries.cluster 0.06206545 0.32234257  
## berry.weight 0.97693622 0.99534625  
## berry.TA 0.70758705 0.92290131  
## berry.pH 0.00364981 0.08204105  
## berry.Brix 0.24722460 0.55823783  
## pruning.weight 0.01701465 0.71830725  
## Ravaz.index 0.52198164 0.89811841

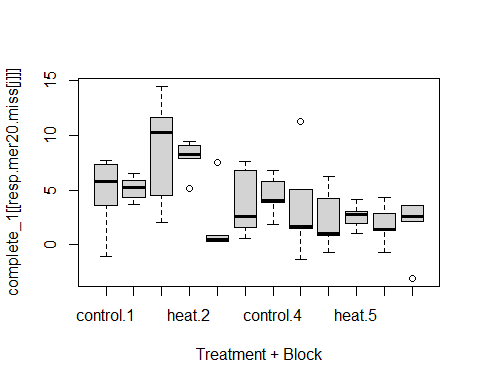
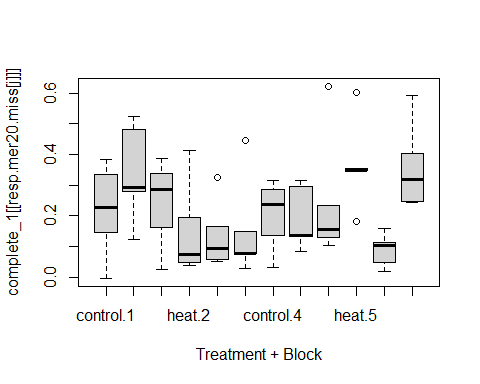
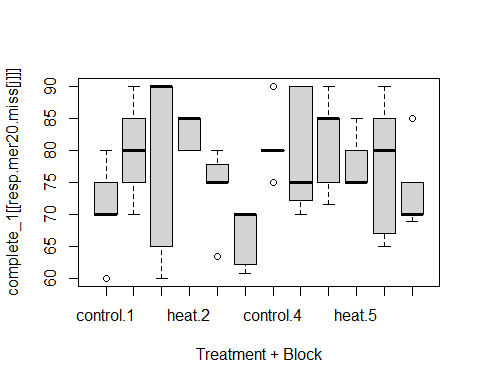
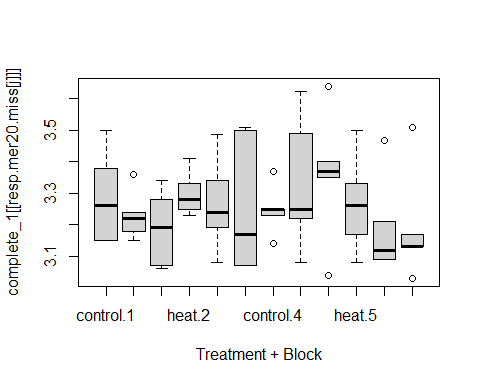
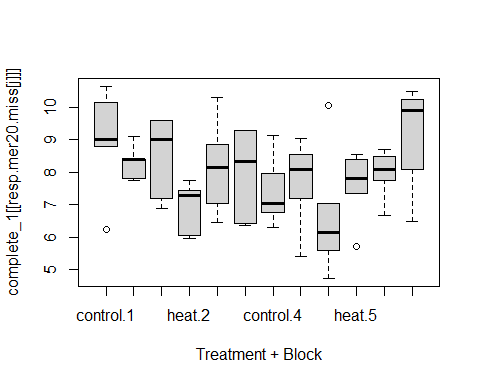
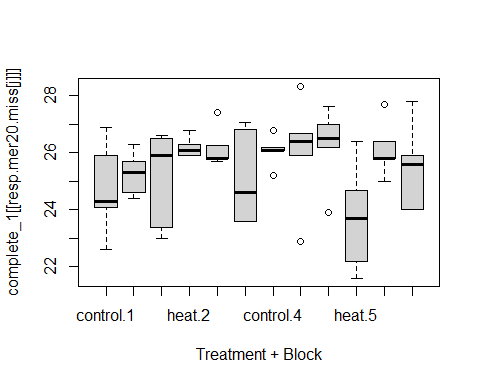
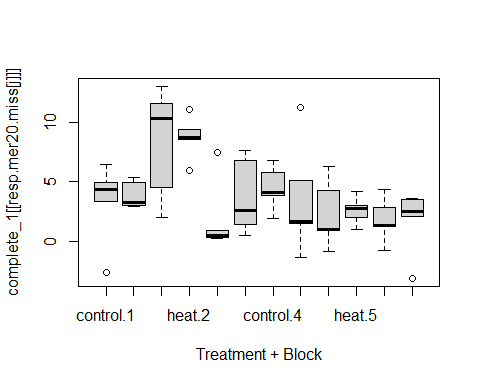
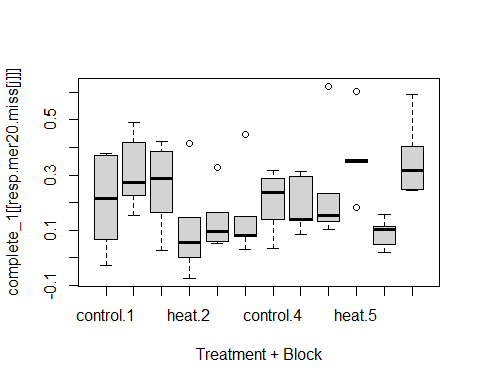
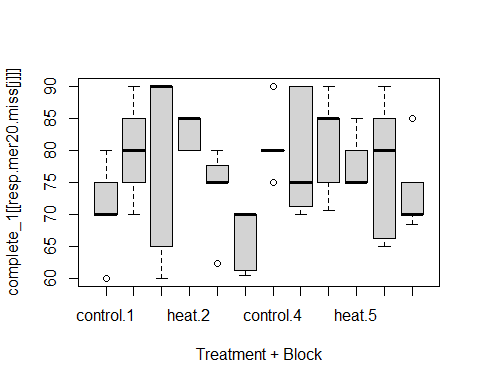
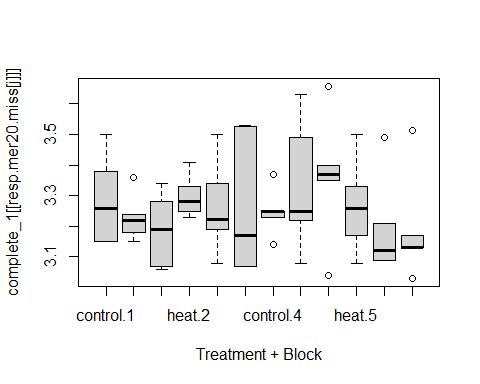
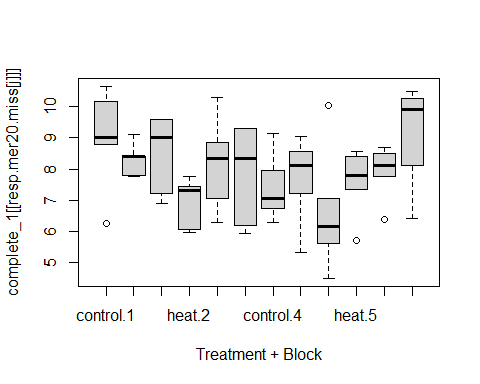
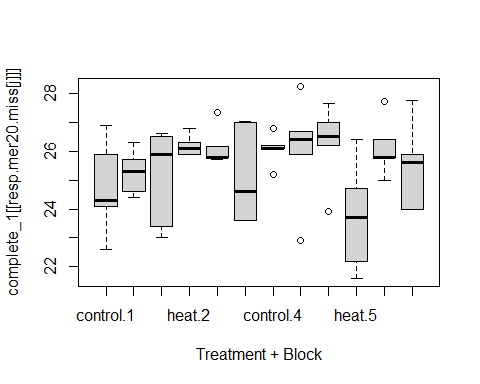
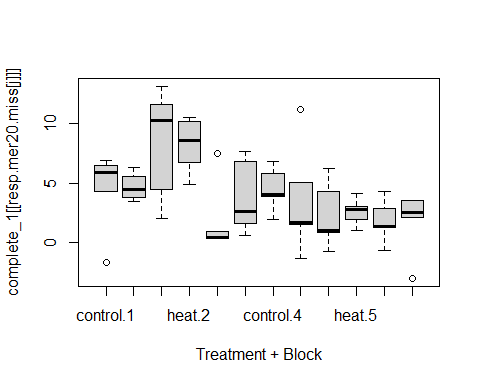
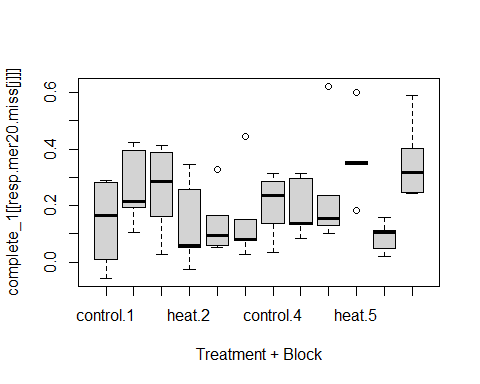
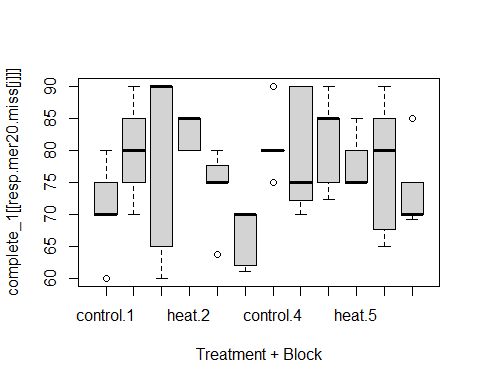
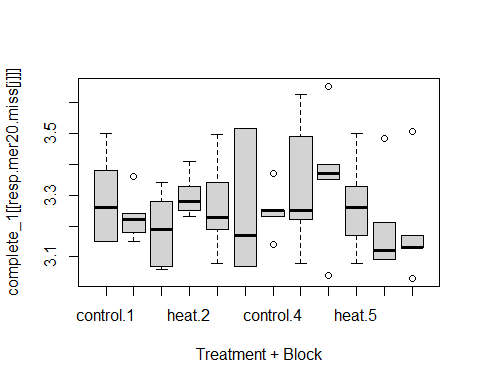
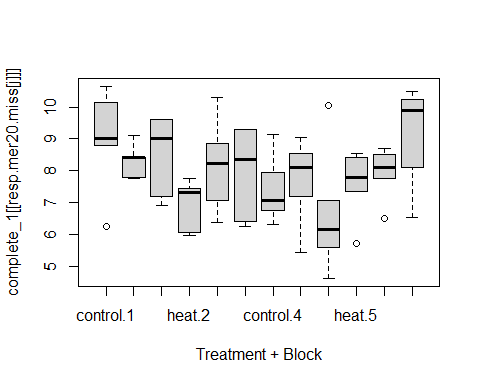
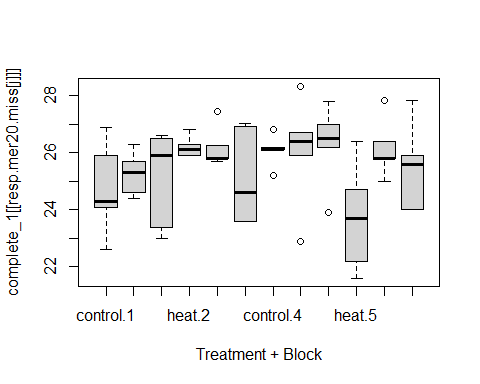
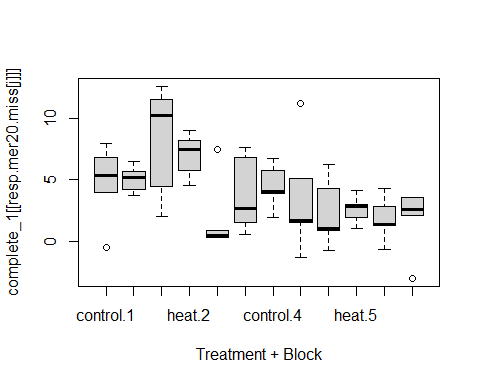
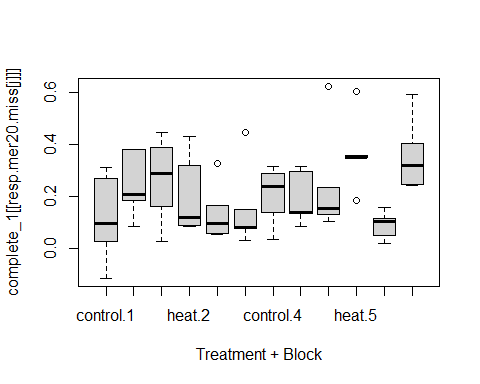
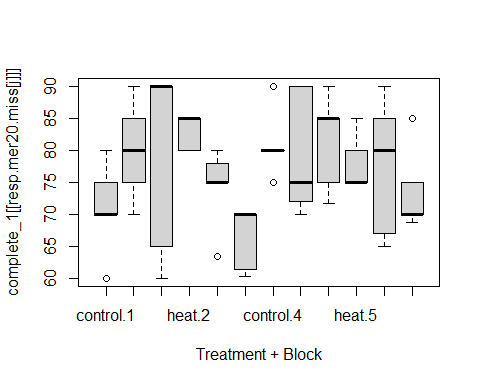
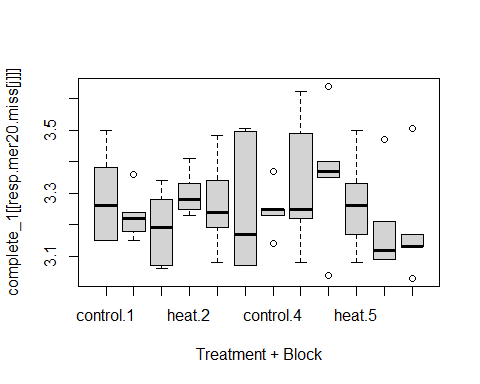
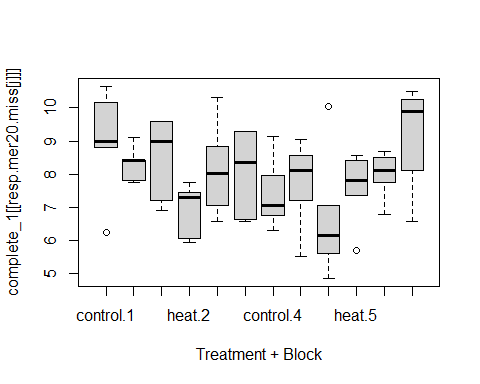
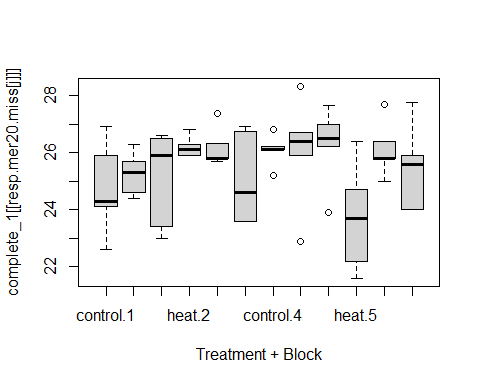
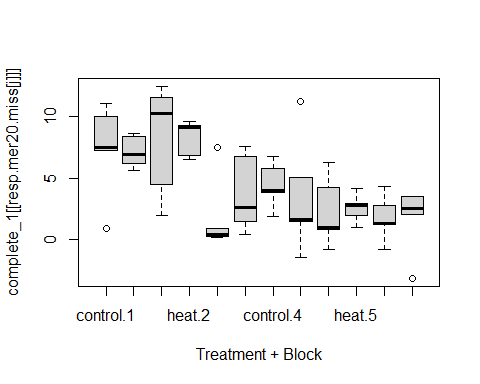
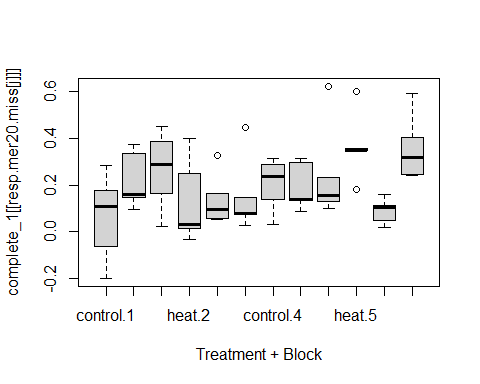
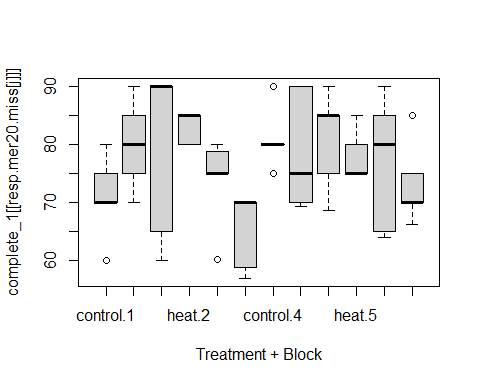
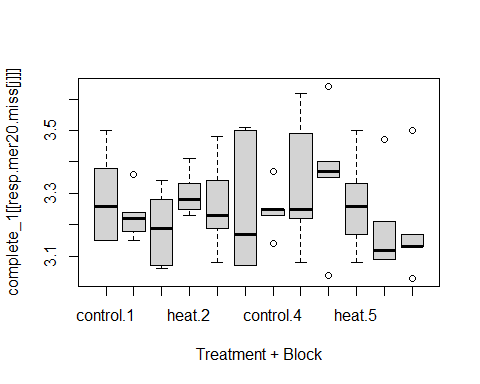
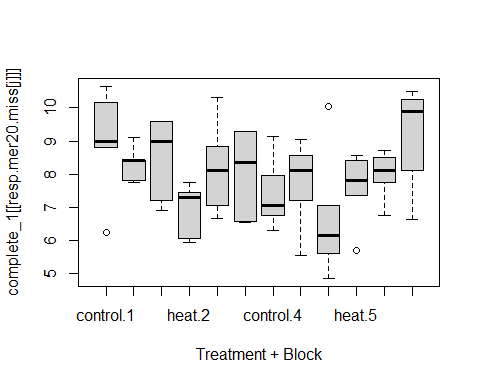
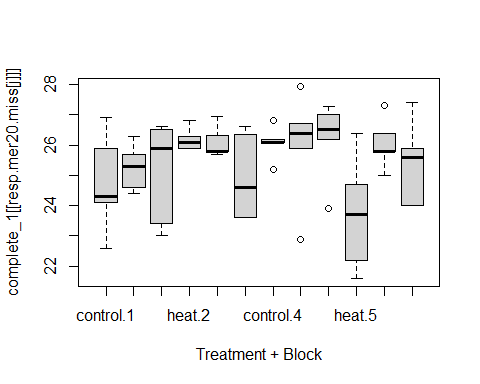
## merlot 2019 normality

# Run shapiro-wilkes tests on each of the variables we will run ANOVAs on  
norm.tests.mer19 <- array(NA, c(12, length(resp.mer19)))  
colnames(norm.tests.mer19) <- resp.mer19  
rownames(norm.tests.mer19) <- unique(paste(merlot2019$block, merlot2019$treatment))[order(unique(paste(merlot2019$block, merlot2019$treatment)))]  
  
# Run the shapiro wilkes test on each of the outcome variables in the Merlot 2019 dataset  
for (i in 1:length(resp.mer19)) {  
 shapiro.results <- with(merlot2019, {lapply(split(eval(parse(text=resp.mer19[i])),   
 paste0(block, treatment)),   
 shapiro.test)})  
 # Save the shapiro wilkes tests for each of the populations (unique combination of block & treatment)  
 # in one large array  
 for (k in 1:12) {norm.tests.mer19[k,i] <- shapiro.results[[k]]$p.value}  
}  
  
# Find the row and column names of the potentially non-normal data  
non.norm <- which(norm.tests.mer19<0.05, arr.ind=T)  
non.norm[,2] <- colnames(norm.tests.mer19)[non.norm[,2]]  
non.norm[,1] <- rownames(norm.tests.mer19)[as.numeric(non.norm[,1])]  
# Plot the Q-Q plots of the potentially non-normal groups  
par(mfrow=c(3,3), mar=c(3,3,3,3))  
for (i in 1:dim(non.norm)[1]) {  
 qqnorm(merlot2019[which(paste(merlot2019$block, merlot2019$treatment)==non.norm[i,1]),non.norm[i,2]],  
 main=paste("Q-Q plot:", non.norm[i,1], non.norm[i,2]))  
 qqline(merlot2019[which(paste(merlot2019$block, merlot2019$treatment)==non.norm[i,1]),non.norm[i,2]])  
}



## merlot 2020 equal variance

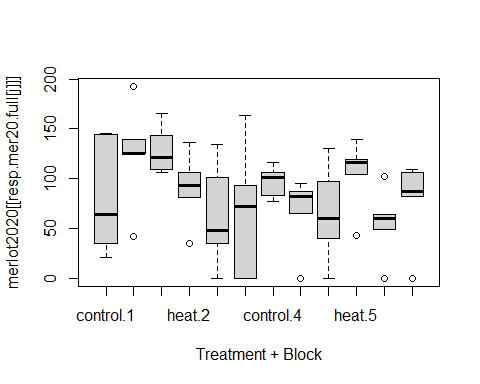
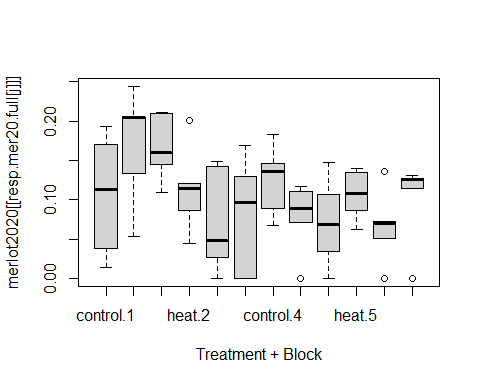
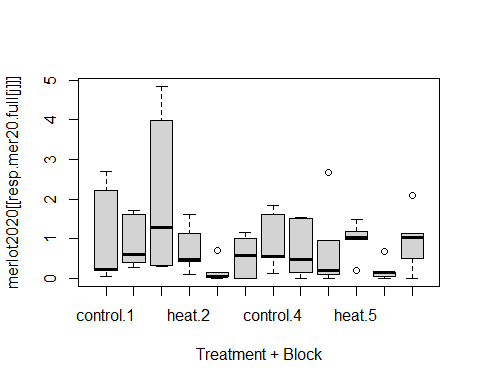
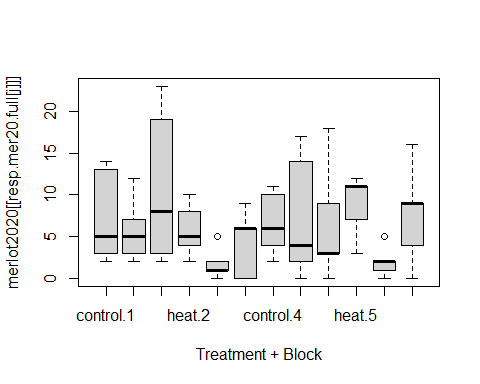
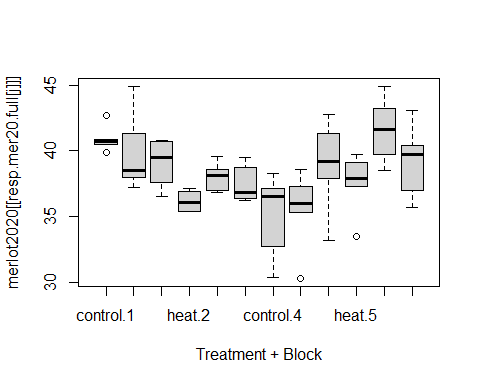
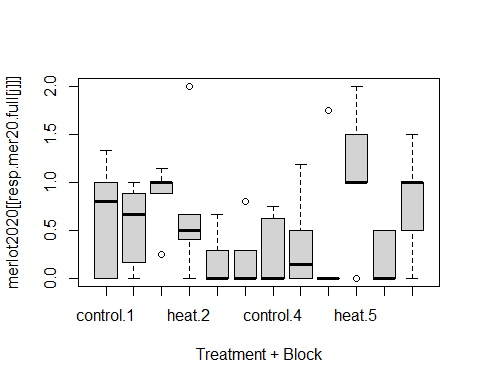
# remove veraison and bloom  
resp.mer20 <- colnames(merlot2020)[-c(1:5, 7,9)]  
  
# responses with missing values  
resp.mer20.miss <- resp.mer20[c(8:13)]  
  
homosc.table.mer20.miss <- array(data=NA, dim=c(length(resp.mer20.miss), 2, 5))  
dimnames(homosc.table.mer20.miss)[[2]] <- c("Bartlett", "Levene")  
dimnames(homosc.table.mer20.miss)[[1]] <- resp.mer20.miss  
  
for (i in 1:length(resp.mer20.miss)) {  
   
 # Run bartlett test on every variable for every  
 # imputed datset  
 bart.p <- unlist(lapply(with(merlot2020mice,   
 bartlett.test(eval(parse(text=resp.mer20.miss[i])) ~ paste(treatment, block)))$analyses,   
 FUN=function(x){x$p.value}))  
   
 # Run levene test on every variable for every imputed  
 # dataset  
 lev.p <- unlist(lapply(with(merlot2020mice,   
 levene.test(eval(parse(text=resp.mer20.miss[i])), paste(treatment, block)))$analyses,  
 FUN=function(x){x$p.value}))   
 homosc.table.mer20.miss[i,1,] <- bart.p  
 homosc.table.mer20.miss[i,2,] <- lev.p  
}  
  
# show plots and table for each imputed datasets  
for (i in 1:5) {  
 complete\_1 = complete(merlot2020mice, i)  
 for (j in 1:length(resp.mer20.miss)) {  
 boxplot(complete\_1[[resp.mer20.miss[j]]] ~ complete\_1$treatment + complete\_1$block,  
 xlab="Treatment + Block")  
 }  
}



homosc.table.mer20.miss

## , , 1  
##   
## Bartlett Levene  
## berry.Brix 0.06038944 0.3495718  
## berry.TA 0.59576856 0.9060027  
## berry.pH 0.47660475 0.8065340  
## brown.seed.color 0.23537231 0.6999803  
## pruning.weight 0.67418390 0.9771337  
## Ravaz.index 0.11499104 0.6926055  
##   
## , , 2  
##   
## Bartlett Levene  
## berry.Brix 0.09228908 0.4111723  
## berry.TA 0.58234662 0.9046463  
## berry.pH 0.47955622 0.8102093  
## brown.seed.color 0.18189154 0.6525656  
## pruning.weight 0.75188867 0.9875096  
## Ravaz.index 0.14038607 0.6888300  
##   
## , , 3  
##   
## Bartlett Levene  
## berry.Brix 0.09787404 0.4200121  
## berry.TA 0.60116749 0.8974638  
## berry.pH 0.44085702 0.8050706  
## brown.seed.color 0.16357623 0.6375287  
## pruning.weight 0.78604317 0.9917968  
## Ravaz.index 0.17614513 0.7483403  
##   
## , , 4  
##   
## Bartlett Levene  
## berry.Brix 0.0870373 0.3863674  
## berry.TA 0.5798118 0.8870790  
## berry.pH 0.4148679 0.7963850  
## brown.seed.color 0.1822882 0.6491871  
## pruning.weight 0.7144605 0.9789867  
## Ravaz.index 0.1338304 0.6927228  
##   
## , , 5  
##   
## Bartlett Levene  
## berry.Brix 0.09070346 0.4034476  
## berry.TA 0.58221061 0.8975786  
## berry.pH 0.47036524 0.8064724  
## brown.seed.color 0.16709993 0.6404652  
## pruning.weight 0.78770751 0.9961960  
## Ravaz.index 0.08868023 0.5512811

# responses without missing values  
resp.mer20.full <- resp.mer20[-c(8:13)]  
  
homosc.table.mer20.full <- array(data=NA, dim=c(length(resp.mer20.full), 2))  
dimnames(homosc.table.mer20.full)[[2]] <- c("Bartlett", "Levene")  
dimnames(homosc.table.mer20.full)[[1]] <- resp.mer20.full  
  
for (i in 1:length(resp.mer20.full)) {  
   
 # Run bartlett test on every variable for every  
 # imputed datset  
 bart.p <- (with(merlot2020,  
 bartlett.test(eval(parse(text=resp.mer20.full[i])) ~ paste(treatment, block)))$p.value)  
   
 # Run levene test on every variable for every imputed  
 # dataset  
 lev.p <- (with(merlot2020,  
 levene.test(eval(parse(text=resp.mer20.full[i])), paste(treatment, block)))$p.value)  
 homosc.table.mer20.full[i,1] <- bart.p  
 homosc.table.mer20.full[i,2] <- lev.p  
}  
  
# show plots and table for each imputed datasets  
for (j in 1:length(resp.mer20.miss)) {  
 boxplot(merlot2020[[resp.mer20.full[j]]] ~ merlot2020$treatment + merlot2020$block,  
 xlab="Treatment + Block")  
}

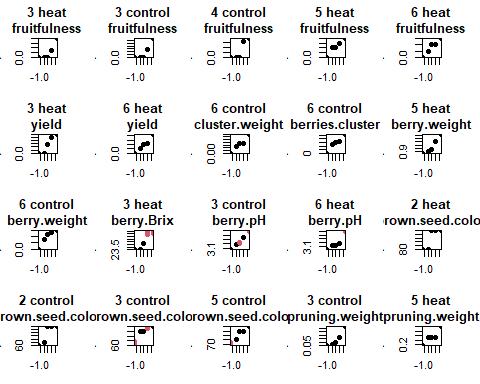


homosc.table.mer20.full

## Bartlett Levene  
## fruitfulness 0.46578082 0.9764925  
## SPAD 0.12816594 0.5998210  
## cluster.number 0.07248268 0.3381961  
## yield 0.00397383 0.1461663  
## cluster.weight 0.91022442 0.8739424  
## berries.cluster 0.52675622 0.6947951  
## berry.weight 0.08241705 0.7434073

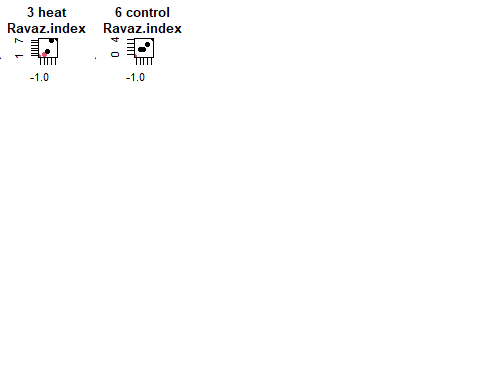
## merlot 2020 normality

# Run shapiro-wilkes tests on each of the variables  
norm.tests.mer20 <- array(NA, c(12, length(resp.mer20), 5))  
dimnames(norm.tests.mer20)[[2]] <- resp.mer20  
dimnames(norm.tests.mer20)[[1]] <- unique(paste(merlot2020$block, merlot2020$treatment))  
  
# Run the shapiro wilkes test on each of the outcome variables in the Merlot 2019 dataset  
for (i in 1:length(resp.mer20)) {  
 shapiro.results <- with(merlot2020mice, {lapply(split(eval(parse(text=resp.mer20[i])),   
 paste0(block, treatment)),   
 shapiro.test)})  
 # Save the shapiro wilkes tests for each of the populations (unique combination of block & treatment)  
 # in one large array  
 for (j in 1:5) {for (k in 1:12) {norm.tests.mer20[k,i,j] <- shapiro.results$analyses[[j]][[k]]$p.value}}}  
  
# Get a table of when a p-value for the Shapiro-Wilkes tests  
# was <0.05, indicating a potential divergence from normality  
nonnorm.mer20 <- apply(norm.tests.mer20, c(1,2), function(x){sum(x<0.05)})  
nonnorm.mer20.rowcol <- which(nonnorm.mer20>0, arr.ind=T)  
nonnorm.mer20.rowcol[,2] <- colnames(nonnorm.mer20)[nonnorm.mer20.rowcol[,2]]  
nonnorm.mer20.rowcol[,1] <- rownames(nonnorm.mer20)[as.numeric(nonnorm.mer20.rowcol[,1])]  
# Plot the Q-Q plots of the potentially non-normal groups  
par(mfrow=c(4,5), mar=c(3,3,3,3))  
for (i in 1:dim(nonnorm.mer20.rowcol)[1]) {  
 dat <- with(merlot2020mice, {  
 dat <- eval(parse(text=nonnorm.mer20.rowcol[i,2]))  
 dat[which(paste(block, treatment)==nonnorm.mer20.rowcol[i,1])]})$analyses  
   
 # Get the Q-Q plot quantiles -- the x-axis of our Q-Q plot  
 quantiles <- rep(c(-1.1797611, -0.4972006, 0, 0.4972006, 1.1797611), 5)  
   
 # The y-values of our imputed datasets, ordered appropriately  
 yvals <- unlist(lapply(dat, function(x){x[order(x)]}))  
   
 # Define the color for the points: the   
 col <- is.na(merlot2020[which(paste(merlot2020$block,   
 merlot2020$treatment)==nonnorm.mer20.rowcol[i,1]),   
 nonnorm.mer20.rowcol[i,2]])[order(dat[[1]])]  
   
 # Plot our Q-Q plot  
 plot(quantiles, yvals, col=rep(col+1, 5),   
 main=paste0(nonnorm.mer20.rowcol[i,1], "\n", nonnorm.mer20.rowcol[i,2]),  
 pch=16)  
}



norm.tests.mer20

## , , 1  
##   
## fruitfulness SPAD cluster.number yield cluster.weight  
## 1 control 0.2840779778 0.2452668 0.1526686 0.05808592 0.521256064  
## 1 heat 0.4214887548 0.3052811 0.5469244 0.13882355 0.469715683  
## 2 heat 0.0571119833 0.3644303 0.2965349 0.16963326 0.446381332  
## 2 control 0.1669046873 0.2284780 0.8584105 0.59739409 0.776903111  
## 3 heat 0.0372931918 0.6709933 0.2229750 0.01156612 0.229091509  
## 3 control 0.0253747605 0.2131361 0.2006238 0.25161962 0.306900434  
## 4 control 0.0206779373 0.4419950 0.6134702 0.26723505 0.818528971  
## 4 heat 0.1170478880 0.3867742 0.2727745 0.13646733 0.223016777  
## 5 heat 0.0001309782 0.7419183 0.2940695 0.06237267 0.964502977  
## 5 control 0.7772534023 0.3058597 0.1936018 0.39238622 0.619068603  
## 6 heat 0.0064700008 0.9046746 0.4531606 0.03065339 0.777069417  
## 6 control 0.8139521274 0.8690318 0.8465565 0.89101631 0.002745906  
## berries.cluster berry.weight berry.Brix berry.TA berry.pH  
## 1 control 0.16712511 0.92152854 0.8657896 0.45175008 0.40813117  
## 1 heat 0.52232992 0.82595575 0.7556224 0.38807145 0.48469351  
## 2 heat 0.45000639 0.45013912 0.1013147 0.10722151 0.44427475  
## 2 control 0.91437687 0.74938361 0.2349601 0.15866448 0.58377064  
## 3 heat 0.81550997 0.55714061 0.1293604 0.77288688 0.94008339  
## 3 control 0.43862797 0.07641954 0.1547327 0.09264184 0.05802472  
## 4 control 0.63398884 0.81306756 0.4339296 0.61659056 0.56355212  
## 4 heat 0.07058167 0.07025726 0.3811091 0.58510732 0.68668769  
## 5 heat 0.98094845 0.01127330 0.1355277 0.35619628 0.55897898  
## 5 control 0.22414641 0.45726254 0.8273279 0.30504936 0.95193860  
## 6 heat 0.72105882 0.30376664 0.8104735 0.54448253 0.04090192  
## 6 control 0.04612380 0.04027945 0.4110110 0.25402588 0.08527598  
## brown.seed.color pruning.weight Ravaz.index  
## 1 control 0.777253402 0.85141843 0.318406522  
## 1 heat 0.967173936 0.24480112 0.431736337  
## 2 heat 0.020677937 0.83764238 0.271554979  
## 2 control 0.006470001 0.27209291 0.096130265  
## 3 heat 0.064269803 0.15163066 0.001243898  
## 3 control 0.017436879 0.04829807 0.307908510  
## 4 control 0.135022591 0.58397731 0.853795945  
## 4 heat 0.077008749 0.16639202 0.511694371  
## 5 heat 0.500607560 0.02749455 0.560470481  
## 5 control 0.045954318 0.29208169 0.968057818  
## 6 heat 0.320652276 0.84377069 0.906447284  
## 6 control 0.329432830 0.25238948 0.024730857  
##   
## , , 2  
##   
## fruitfulness SPAD cluster.number yield cluster.weight  
## 1 control 0.2840779778 0.2452668 0.1526686 0.05808592 0.521256064  
## 1 heat 0.4214887548 0.3052811 0.5469244 0.13882355 0.469715683  
## 2 heat 0.0571119833 0.3644303 0.2965349 0.16963326 0.446381332  
## 2 control 0.1669046873 0.2284780 0.8584105 0.59739409 0.776903111  
## 3 heat 0.0372931918 0.6709933 0.2229750 0.01156612 0.229091509  
## 3 control 0.0253747605 0.2131361 0.2006238 0.25161962 0.306900434  
## 4 control 0.0206779373 0.4419950 0.6134702 0.26723505 0.818528971  
## 4 heat 0.1170478880 0.3867742 0.2727745 0.13646733 0.223016777  
## 5 heat 0.0001309782 0.7419183 0.2940695 0.06237267 0.964502977  
## 5 control 0.7772534023 0.3058597 0.1936018 0.39238622 0.619068603  
## 6 heat 0.0064700008 0.9046746 0.4531606 0.03065339 0.777069417  
## 6 control 0.8139521274 0.8690318 0.8465565 0.89101631 0.002745906  
## berries.cluster berry.weight berry.Brix berry.TA berry.pH  
## 1 control 0.16712511 0.92152854 0.86578964 0.4517501 0.40813117  
## 1 heat 0.52232992 0.82595575 0.75562242 0.3880714 0.48469351  
## 2 heat 0.45000639 0.45013912 0.10131466 0.1072215 0.44427475  
## 2 control 0.91437687 0.74938361 0.23496015 0.1586645 0.58377064  
## 3 heat 0.81550997 0.55714061 0.05914246 0.8045705 0.96930131  
## 3 control 0.43862797 0.07641954 0.11338116 0.1058624 0.05814054  
## 4 control 0.63398884 0.81306756 0.43392964 0.6165906 0.56355212  
## 4 heat 0.07058167 0.07025726 0.53575695 0.5683143 0.68921132  
## 5 heat 0.98094845 0.01127330 0.35808627 0.3555662 0.55743710  
## 5 control 0.22414641 0.45726254 0.82732787 0.3050494 0.95193860  
## 6 heat 0.72105882 0.30376664 0.61588617 0.5701826 0.04156705  
## 6 control 0.04612380 0.04027945 0.38542280 0.2518528 0.08073056  
## brown.seed.color pruning.weight Ravaz.index  
## 1 control 0.777253402 0.72066970 0.537584550  
## 1 heat 0.967173936 0.33584523 0.856998757  
## 2 heat 0.020677937 0.79458534 0.303456452  
## 2 control 0.006470001 0.12819784 0.781172106  
## 3 heat 0.144231569 0.15163066 0.001056796  
## 3 control 0.016117908 0.04829807 0.297672560  
## 4 control 0.135022591 0.58397731 0.853795945  
## 4 heat 0.087250005 0.16639202 0.496488111  
## 5 heat 0.437075520 0.02749455 0.540979623  
## 5 control 0.045954318 0.29208169 0.968057818  
## 6 heat 0.438405759 0.84377069 0.906019451  
## 6 control 0.080108922 0.25238948 0.025677658  
##   
## , , 3  
##   
## fruitfulness SPAD cluster.number yield cluster.weight  
## 1 control 0.2840779778 0.2452668 0.1526686 0.05808592 0.521256064  
## 1 heat 0.4214887548 0.3052811 0.5469244 0.13882355 0.469715683  
## 2 heat 0.0571119833 0.3644303 0.2965349 0.16963326 0.446381332  
## 2 control 0.1669046873 0.2284780 0.8584105 0.59739409 0.776903111  
## 3 heat 0.0372931918 0.6709933 0.2229750 0.01156612 0.229091509  
## 3 control 0.0253747605 0.2131361 0.2006238 0.25161962 0.306900434  
## 4 control 0.0206779373 0.4419950 0.6134702 0.26723505 0.818528971  
## 4 heat 0.1170478880 0.3867742 0.2727745 0.13646733 0.223016777  
## 5 heat 0.0001309782 0.7419183 0.2940695 0.06237267 0.964502977  
## 5 control 0.7772534023 0.3058597 0.1936018 0.39238622 0.619068603  
## 6 heat 0.0064700008 0.9046746 0.4531606 0.03065339 0.777069417  
## 6 control 0.8139521274 0.8690318 0.8465565 0.89101631 0.002745906  
## berries.cluster berry.weight berry.Brix berry.TA berry.pH  
## 1 control 0.16712511 0.92152854 0.86578964 0.4517501 0.40813117  
## 1 heat 0.52232992 0.82595575 0.75562242 0.3880714 0.48469351  
## 2 heat 0.45000639 0.45013912 0.10131466 0.1072215 0.44427475  
## 2 control 0.91437687 0.74938361 0.23496015 0.1586645 0.58377064  
## 3 heat 0.81550997 0.55714061 0.03550438 0.9146712 0.90425200  
## 3 control 0.43862797 0.07641954 0.09191079 0.1168531 0.04958182  
## 4 control 0.63398884 0.81306756 0.43392964 0.6165906 0.56355212  
## 4 heat 0.07058167 0.07025726 0.53811468 0.5386642 0.69252310  
## 5 heat 0.98094845 0.01127330 0.44021161 0.4833667 0.57044244  
## 5 control 0.22414641 0.45726254 0.82732787 0.3050494 0.95193860  
## 6 heat 0.72105882 0.30376664 0.53658525 0.3816100 0.03607728  
## 6 control 0.04612380 0.04027945 0.37794221 0.2496333 0.07858573  
## brown.seed.color pruning.weight Ravaz.index  
## 1 control 0.777253402 0.33049509 0.0606650492  
## 1 heat 0.967173936 0.40444267 0.6180234326  
## 2 heat 0.020677937 0.58580202 0.4063448465  
## 2 control 0.006470001 0.40293556 0.5723986684  
## 3 heat 0.160239889 0.15163066 0.0009904835  
## 3 control 0.014073635 0.04829807 0.2959137511  
## 4 control 0.135022591 0.58397731 0.8537959448  
## 4 heat 0.089435679 0.16639202 0.4935538915  
## 5 heat 0.404731704 0.02749455 0.5345061087  
## 5 control 0.045954318 0.29208169 0.9680578176  
## 6 heat 0.508246064 0.84377069 0.9061606260  
## 6 control 0.051746809 0.25238948 0.0260067285  
##   
## , , 4  
##   
## fruitfulness SPAD cluster.number yield cluster.weight  
## 1 control 0.2840779778 0.2452668 0.1526686 0.05808592 0.521256064  
## 1 heat 0.4214887548 0.3052811 0.5469244 0.13882355 0.469715683  
## 2 heat 0.0571119833 0.3644303 0.2965349 0.16963326 0.446381332  
## 2 control 0.1669046873 0.2284780 0.8584105 0.59739409 0.776903111  
## 3 heat 0.0372931918 0.6709933 0.2229750 0.01156612 0.229091509  
## 3 control 0.0253747605 0.2131361 0.2006238 0.25161962 0.306900434  
## 4 control 0.0206779373 0.4419950 0.6134702 0.26723505 0.818528971  
## 4 heat 0.1170478880 0.3867742 0.2727745 0.13646733 0.223016777  
## 5 heat 0.0001309782 0.7419183 0.2940695 0.06237267 0.964502977  
## 5 control 0.7772534023 0.3058597 0.1936018 0.39238622 0.619068603  
## 6 heat 0.0064700008 0.9046746 0.4531606 0.03065339 0.777069417  
## 6 control 0.8139521274 0.8690318 0.8465565 0.89101631 0.002745906  
## berries.cluster berry.weight berry.Brix berry.TA berry.pH  
## 1 control 0.16712511 0.92152854 0.86578964 0.4517501 0.40813117  
## 1 heat 0.52232992 0.82595575 0.75562242 0.3880714 0.48469351  
## 2 heat 0.45000639 0.45013912 0.10131466 0.1072215 0.44427475  
## 2 control 0.91437687 0.74938361 0.23496015 0.1586645 0.58377064  
## 3 heat 0.81550997 0.55714061 0.02346822 0.9317635 0.86844989  
## 3 control 0.43862797 0.07641954 0.07990362 0.1182110 0.04966206  
## 4 control 0.63398884 0.81306756 0.43392964 0.6165906 0.56355212  
## 4 heat 0.07058167 0.07025726 0.50838131 0.4973448 0.69467627  
## 5 heat 0.98094845 0.01127330 0.34830855 0.5460219 0.57281806  
## 5 control 0.22414641 0.45726254 0.82732787 0.3050494 0.95193860  
## 6 heat 0.72105882 0.30376664 0.60277562 0.3147892 0.03485878  
## 6 control 0.04612380 0.04027945 0.38665691 0.2411206 0.07495188  
## brown.seed.color pruning.weight Ravaz.index  
## 1 control 0.777253402 0.40365841 0.179972024  
## 1 heat 0.967173936 0.71750463 0.097826232  
## 2 heat 0.020677937 0.64937732 0.378142292  
## 2 control 0.006470001 0.45523283 0.741182750  
## 3 heat 0.114162530 0.15163066 0.001300177  
## 3 control 0.012702585 0.04829807 0.303274077  
## 4 control 0.135022591 0.58397731 0.853795945  
## 4 heat 0.076029280 0.16639202 0.508902376  
## 5 heat 0.475669560 0.02749455 0.564153983  
## 5 control 0.045954318 0.29208169 0.968057818  
## 6 heat 0.361805896 0.84377069 0.906583285  
## 6 control 0.097906146 0.25238948 0.024818347  
##   
## , , 5  
##   
## fruitfulness SPAD cluster.number yield cluster.weight  
## 1 control 0.2840779778 0.2452668 0.1526686 0.05808592 0.521256064  
## 1 heat 0.4214887548 0.3052811 0.5469244 0.13882355 0.469715683  
## 2 heat 0.0571119833 0.3644303 0.2965349 0.16963326 0.446381332  
## 2 control 0.1669046873 0.2284780 0.8584105 0.59739409 0.776903111  
## 3 heat 0.0372931918 0.6709933 0.2229750 0.01156612 0.229091509  
## 3 control 0.0253747605 0.2131361 0.2006238 0.25161962 0.306900434  
## 4 control 0.0206779373 0.4419950 0.6134702 0.26723505 0.818528971  
## 4 heat 0.1170478880 0.3867742 0.2727745 0.13646733 0.223016777  
## 5 heat 0.0001309782 0.7419183 0.2940695 0.06237267 0.964502977  
## 5 control 0.7772534023 0.3058597 0.1936018 0.39238622 0.619068603  
## 6 heat 0.0064700008 0.9046746 0.4531606 0.03065339 0.777069417  
## 6 control 0.8139521274 0.8690318 0.8465565 0.89101631 0.002745906  
## berries.cluster berry.weight berry.Brix berry.TA berry.pH  
## 1 control 0.16712511 0.92152854 0.86578964 0.4517501 0.40813117  
## 1 heat 0.52232992 0.82595575 0.75562242 0.3880714 0.48469351  
## 2 heat 0.45000639 0.45013912 0.10131466 0.1072215 0.44427475  
## 2 control 0.91437687 0.74938361 0.23496015 0.1586645 0.58377064  
## 3 heat 0.81550997 0.55714061 0.03944652 0.8862067 0.96336279  
## 3 control 0.43862797 0.07641954 0.11285100 0.0993090 0.05808920  
## 4 control 0.63398884 0.81306756 0.43392964 0.6165906 0.56355212  
## 4 heat 0.07058167 0.07025726 0.54138312 0.5297101 0.69093427  
## 5 heat 0.98094845 0.01127330 0.34530904 0.4216436 0.55753125  
## 5 control 0.22414641 0.45726254 0.82732787 0.3050494 0.95193860  
## 6 heat 0.72105882 0.30376664 0.60681092 0.4801167 0.04202350  
## 6 control 0.04612380 0.04027945 0.38001426 0.2469990 0.07847995  
## brown.seed.color pruning.weight Ravaz.index  
## 1 control 0.777253402 0.82538892 0.2802379216  
## 1 heat 0.967173936 0.58076123 0.9275286601  
## 2 heat 0.020677937 0.64072425 0.6451813530  
## 2 control 0.006470001 0.09856223 0.2435509925  
## 3 heat 0.149811205 0.15163066 0.0009974983  
## 3 control 0.018770986 0.04829807 0.2958243987  
## 4 control 0.135022591 0.58397731 0.8537959448  
## 4 heat 0.089521311 0.16639202 0.4974214980  
## 5 heat 0.443872324 0.02749455 0.5406081296  
## 5 control 0.045954318 0.29208169 0.9680578176  
## 6 heat 0.441848039 0.84377069 0.9059260370  
## 6 control 0.062576026 0.25238948 0.0254740683



# Two-way ANOVA and ART ANOVA

## chardonnay 2019

# table keeping track of which effect is significant  
char.2019.sig.anova <- array(data=rep(0, length(resp.char19) \* 3), dim=c(length(resp.char19), 3))  
dimnames(char.2019.sig.anova)[[2]] <- c("treatment", "block", "interaction")  
dimnames(char.2019.sig.anova)[[1]] <- resp.char19  
  
char.2019.sig.art <- array(data=rep(0, length(resp.char19) \* 3), dim=c(length(resp.char19), 3))  
dimnames(char.2019.sig.art)[[2]] <- c("treatment", "block", "interaction")  
dimnames(char.2019.sig.art)[[1]] <- resp.char19

resps = colnames(chardonnay2019)[6:8]  
for (i in 1:length(resps)) {  
 print(resps[i])  
   
 res = aov( eval(parse(text=resps[i])) ~ treatment + block + treatment:block, chardonnay2019)  
 p.val = summary(res)[[1]][["Pr(>F)"]][c(1:3)]  
 char.2019.sig.anova[i,] = p.val  
   
 print(summary(res))  
   
 model = art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, chardonnay2019)  
 res = anova(model, type = 1)  
 p.val = res[["Pr(>F)"]][c(1:3)]  
 char.2019.sig.art[i,] = p.val  
   
 print(res)  
}

## [1] "X..of.clusters"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 55.2 55.22 1.176 0.287  
## block 4 285.2 71.29 1.518 0.222  
## treatment:block 4 322.6 80.66 1.718 0.172  
## Residuals 30 1408.8 46.96   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 30 1.2226 0.277632   
## 2 block 4 30 2.4189 0.070334 .  
## 3 treatment:block 4 30 2.4056 0.071534 .  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "yield..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 1.45 1.452 0.373 0.546  
## block 4 20.44 5.111 1.312 0.288  
## treatment:block 4 0.03 0.007 0.002 1.000  
## Residuals 30 116.85 3.895   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 30 0.991545 0.32733   
## 2 block 4 30 1.355386 0.27276   
## 3 treatment:block 4 30 0.046131 0.99576   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "av.cluster.weight..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.00001 0.0000057 0.003 0.956  
## block 4 0.00716 0.0017912 0.964 0.441  
## treatment:block 4 0.01036 0.0025897 1.394 0.260  
## Residuals 30 0.05572 0.0018573   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 30 0.00902 0.92497   
## 2 block 4 30 0.64523 0.63454   
## 3 treatment:block 4 30 1.36546 0.26932   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

print(char.2019.sig.anova)

## treatment block interaction  
## X..of.clusters 0.2868016 0.2220049 0.1721043  
## yield..kg. 0.5461290 0.2880019 0.9999938  
## av.cluster.weight..kg. 0.9562280 0.4413318 0.2596908

print(char.2019.sig.art)

## treatment block interaction  
## X..of.clusters 0.2776317 0.07033402 0.07153366  
## yield..kg. 0.3273293 0.27275716 0.99576391  
## av.cluster.weight..kg. 0.9249672 0.63454452 0.26932026

char.2019.sig = array(data=rep(0, length(resp.char19) \* 2), dim=c(length(resp.char19), 2))  
dimnames(char.2019.sig)[[2]] <- c("treatment", "interaction")  
dimnames(char.2019.sig)[[1]] <- resp.char19  
for (i in 1:length(resp.char19)) {  
 for (j in c(1,3)) {  
 if (j == 1) {  
 char.2019.sig[i,j] = max(char.2019.sig.anova[i,j], char.2019.sig.art[i,j])  
 char.2019.sig[i,j] = (char.2019.sig[i,j] <= 0.05)   
 } else {  
 char.2019.sig[i,j-1] = max(char.2019.sig.anova[i,j], char.2019.sig.art[i,j])  
 char.2019.sig[i,j-1] = (char.2019.sig[i,j-1] <= 0.05)   
 }  
 }  
}  
print(char.2019.sig)

## treatment interaction  
## X..of.clusters 0 0  
## yield..kg. 0 0  
## av.cluster.weight..kg. 0 0

## chardonnay 2020

# table keeping track of which effect is significant  
char.2020.sig.anova <- array(data=rep(0, length(resp.char20) \* 3), dim=c(length(resp.char20), 3))  
dimnames(char.2020.sig.anova)[[2]] <- c("treatment", "block", "interaction")  
dimnames(char.2020.sig.anova)[[1]] <- resp.char20  
  
char.2020.sig.art <- array(data=rep(0, length(resp.char20) \* 3), dim=c(length(resp.char20), 3))  
dimnames(char.2020.sig.art)[[2]] <- c("treatment", "block", "interaction")  
dimnames(char.2020.sig.art)[[1]] <- resp.char20

resps = colnames(chardonnay2020)[6:8]  
for (i in 1:length(resps)) {  
 print(resps[i])  
   
 res = aov( eval(parse(text=resps[i])) ~ treatment + block + treatment:block, chardonnay2020)  
 p.val = summary(res)[[1]][["Pr(>F)"]][c(1:3)]  
 char.2020.sig.anova[i,] = p.val  
   
 print(summary(res))  
   
 model = art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, chardonnay2020)  
 res = anova(model, type = 1)  
 p.val = res[["Pr(>F)"]][c(1:3)]  
 char.2020.sig.art[i,] = p.val  
   
 print(res)  
}

## [1] "X..of.clusters"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 72.9 72.90 1.181 0.2859   
## block 4 690.6 172.65 2.796 0.0438 \*  
## treatment:block 4 499.6 124.90 2.023 0.1165   
## Residuals 30 1852.5 61.75   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 30 1.0012 0.325025   
## 2 block 4 30 2.9615 0.035634 \*  
## 3 treatment:block 4 30 2.8050 0.043283 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "yield..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.54 0.541 0.133 0.7182   
## block 4 53.14 13.285 3.256 0.0248 \*  
## treatment:block 4 20.75 5.188 1.271 0.3031   
## Residuals 30 122.41 4.080   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 30 0.11283 0.739286   
## 2 block 4 30 2.99336 0.034257 \*  
## 3 treatment:block 4 30 1.50554 0.225576   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "av.cluster.weight..kg."  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.00190 0.001902 0.824 0.3712   
## block 4 0.02151 0.005377 2.329 0.0788 .  
## treatment:block 4 0.00310 0.000775 0.336 0.8516   
## Residuals 30 0.06925 0.002308   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 30 0.71637 0.404037   
## 2 block 4 30 2.45415 0.067265 .  
## 3 treatment:block 4 30 0.36349 0.832608   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

print(char.2020.sig.anova)

## treatment block interaction  
## X..of.clusters 0.2858941 0.04377424 0.1165051  
## yield..kg. 0.7182004 0.02482398 0.3030703  
## av.cluster.weight..kg. 0.3712009 0.07879033 0.8516140

print(char.2020.sig.art)

## treatment block interaction  
## X..of.clusters 0.3250253 0.03563390 0.04328331  
## yield..kg. 0.7392859 0.03425659 0.22557578  
## av.cluster.weight..kg. 0.4040369 0.06726486 0.83260768

char.2020.sig = array(data=rep(0, length(resp.char20) \* 2), dim=c(length(resp.char20), 2))  
dimnames(char.2020.sig)[[2]] <- c("treatment", "interaction")  
dimnames(char.2020.sig)[[1]] <- resp.char20  
for (i in 1:length(resp.char20)) {  
 for (j in c(1,3)) {  
 if (j == 1) {  
 char.2020.sig[i,j] = max(char.2020.sig.anova[i,j], char.2020.sig.art[i,j])  
 char.2020.sig[i,j] = (char.2020.sig[i,j] <= 0.05)   
 } else {  
 char.2020.sig[i,j-1] = max(char.2020.sig.anova[i,j], char.2020.sig.art[i,j])  
 char.2020.sig[i,j-1] = (char.2020.sig[i,j-1] <= 0.05)   
 }  
 }  
}  
print(char.2020.sig)

## treatment interaction  
## X..of.clusters 0 0  
## yield..kg. 0 0  
## av.cluster.weight..kg. 0 0

## merlot 2019

# table keeping track of which effect is significant  
mer.2019.sig.anova <- array(data=rep(0, length(resp.mer19) \* 3), dim=c(length(resp.mer19), 3))  
dimnames(mer.2019.sig.anova)[[2]] <- c("treatment", "block", "interaction")  
dimnames(mer.2019.sig.anova)[[1]] <- resp.mer19  
  
mer.2019.sig.art <- array(data=rep(0, length(resp.mer19) \* 3), dim=c(length(resp.mer19), 3))  
dimnames(mer.2019.sig.art)[[2]] <- c("treatment", "block", "interaction")  
dimnames(mer.2019.sig.art)[[1]] <- resp.mer19

resps = resp.mer19  
for (i in 1:length(resps)) {  
 print(resps[i])  
   
 res = aov( eval(parse(text=resps[i])) ~ treatment + block + treatment:block, merlot2019)  
 p.val = summary(res)[[1]][["Pr(>F)"]][c(1:3)]  
 mer.2019.sig.anova[i,] = p.val  
   
 print(summary(res))  
   
 model = art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, merlot2019)  
 res = anova(model, type = 1)  
 p.val = res[["Pr(>F)"]][c(1:3)]  
 mer.2019.sig.art[i,] = p.val  
   
 print(res)  
}

## [1] "SPAD"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 4.82 4.817 1.265 0.266270   
## block 5 99.92 19.985 5.249 0.000635 \*\*\*  
## treatment:block 5 15.86 3.171 0.833 0.532808   
## Residuals 48 182.74 3.807   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 1.23468 0.27203346   
## 2 block 5 48 5.15877 0.00072398 \*\*\*  
## 3 treatment:block 5 48 0.68572 0.63653818   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "cluster.number"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 36.8 36.82 1.183 0.282  
## block 5 145.5 29.11 0.935 0.467  
## treatment:block 5 18.1 3.62 0.116 0.988  
## Residuals 48 1494.4 31.13   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 1.14645 0.28965   
## 2 block 5 48 1.09302 0.37648   
## 3 treatment:block 5 48 0.14879 0.97945   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "yield"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 3.77 3.769 3.149 0.08233 .   
## block 5 21.03 4.205 3.513 0.00871 \*\*  
## treatment:block 5 7.71 1.542 1.288 0.28455   
## Residuals 48 57.46 1.197   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 1.75386 0.191664   
## 2 block 5 48 3.30331 0.012106 \*  
## 3 treatment:block 5 48 0.89071 0.494812   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "cluster.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.00278 0.002775 1.434 0.236980   
## block 5 0.04879 0.009757 5.042 0.000859 \*\*\*  
## treatment:block 5 0.02179 0.004359 2.252 0.064080 .   
## Residuals 48 0.09289 0.001935   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 1.7599 0.1909149   
## 2 block 5 48 4.8313 0.0011712 \*\*  
## 3 treatment:block 5 48 2.4133 0.0496174 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berries.cluster"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 2809 2808.8 3.603 0.06369 .   
## block 5 15480 3096.0 3.972 0.00428 \*\*  
## treatment:block 5 9454 1890.8 2.426 0.04866 \*   
## Residuals 48 37417 779.5   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 3.9742 0.0519032 .  
## 2 block 5 48 4.1719 0.0031532 \*\*  
## 3 treatment:block 5 48 2.3425 0.0555391 .  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berry.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.0338 0.03384 1.731 0.19459   
## block 5 0.4171 0.08342 4.266 0.00273 \*\*  
## treatment:block 5 0.4055 0.08110 4.147 0.00328 \*\*  
## Residuals 48 0.9387 0.01956   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 1.3632 0.2487522   
## 2 block 5 48 3.6486 0.0070564 \*\*  
## 3 treatment:block 5 48 3.7140 0.0063749 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berry.TA"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.62 0.621 0.485 0.4896   
## block 5 18.36 3.673 2.867 0.0241 \*  
## treatment:block 5 6.25 1.250 0.975 0.4425   
## Residuals 48 61.50 1.281   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.65214 0.423331   
## 2 block 5 48 2.79791 0.026914 \*  
## 3 treatment:block 5 48 0.99728 0.429536   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berry.pH"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.01441 0.014415 3.085 0.085392 .   
## block 5 0.15320 0.030639 6.557 0.000101 \*\*\*  
## treatment:block 5 0.08060 0.016119 3.450 0.009622 \*\*   
## Residuals 48 0.22428 0.004673   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 4.8498 0.03248675 \*  
## 2 block 5 48 5.9388 0.00023712 \*\*\*  
## 3 treatment:block 5 48 4.2555 0.00277690 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berry.Brix"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.662 0.6615 1.341 0.253  
## block 5 4.751 0.9502 1.926 0.107  
## treatment:block 5 1.454 0.2907 0.589 0.708  
## Residuals 48 23.684 0.4934   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.30407 0.58390   
## 2 block 5 48 1.56051 0.18926   
## 3 treatment:block 5 48 0.41667 0.83484   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "pruning.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.0013 0.00131 0.063 0.8025   
## block 5 0.2661 0.05322 2.578 0.0382 \*  
## treatment:block 5 0.2864 0.05728 2.774 0.0279 \*  
## Residuals 48 0.9911 0.02065   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.022489 0.881423   
## 2 block 5 48 2.933237 0.021713 \*  
## 3 treatment:block 5 48 3.355536 0.011152 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "Ravaz.index"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 98.0 98.01 2.694 0.10727   
## block 5 185.9 37.19 1.022 0.41527   
## treatment:block 5 651.8 130.36 3.583 0.00781 \*\*  
## Residuals 48 1746.3 36.38   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 1.9848 0.165327   
## 2 block 5 48 1.1692 0.338058   
## 3 treatment:block 5 48 3.5865 0.007772 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

print(mer.2019.sig.anova)

## treatment block interaction  
## SPAD 0.26627032 0.0006347289 0.532807678  
## cluster.number 0.28226834 0.4668984801 0.988223756  
## yield 0.08232938 0.0087125437 0.284550573  
## cluster.weight 0.23698050 0.0008586905 0.064080212  
## berries.cluster 0.06368702 0.0042828853 0.048663366  
## berry.weight 0.19459497 0.0027348384 0.003277064  
## berry.TA 0.48961275 0.0241349880 0.442478427  
## berry.pH 0.08539214 0.0001007907 0.009621749  
## berry.Brix 0.25264803 0.1074027551 0.708191435  
## pruning.weight 0.80245199 0.0382081222 0.027941093  
## Ravaz.index 0.10726725 0.4152650314 0.007812617

print(mer.2019.sig.art)

## treatment block interaction  
## SPAD 0.27203346 0.0007239844 0.636538179  
## cluster.number 0.28964756 0.3764754757 0.979454661  
## yield 0.19166421 0.0121055535 0.494811972  
## cluster.weight 0.19091485 0.0011712346 0.049617428  
## berries.cluster 0.05190325 0.0031532371 0.055539074  
## berry.weight 0.24875216 0.0070564180 0.006374889  
## berry.TA 0.42333097 0.0269140647 0.429535516  
## berry.pH 0.03248675 0.0002371157 0.002776901  
## berry.Brix 0.58389836 0.1892568398 0.834837495  
## pruning.weight 0.88142293 0.0217128611 0.011152410  
## Ravaz.index 0.16532747 0.3380581008 0.007772031

mer.2019.sig = array(data=rep(0, length(resp.mer19) \* 2), dim=c(length(resp.mer19), 2))  
dimnames(mer.2019.sig)[[2]] <- c("treatment", "interaction")  
dimnames(mer.2019.sig)[[1]] <- resp.mer19  
for (i in 1:length(resp.mer19)) {  
 for (j in c(1,3)) {  
 if (j == 1) {  
 mer.2019.sig[i,j] = max(mer.2019.sig.anova[i,j], mer.2019.sig.art[i,j])  
 mer.2019.sig[i,j] = (mer.2019.sig[i,j] <= 0.05)   
 } else {  
 mer.2019.sig[i,j-1] = max(mer.2019.sig.anova[i,j], mer.2019.sig.art[i,j])  
 mer.2019.sig[i,j-1] = (mer.2019.sig[i,j-1] <= 0.05)   
 }  
 }  
}  
print(mer.2019.sig)

## treatment interaction  
## SPAD 0 0  
## cluster.number 0 0  
## yield 0 0  
## cluster.weight 0 0  
## berries.cluster 0 0  
## berry.weight 0 1  
## berry.TA 0 0  
## berry.pH 0 1  
## berry.Brix 0 0  
## pruning.weight 0 1  
## Ravaz.index 0 1

## merlot 2020

# table keeping track of which effect is significant  
mer.2020.sig.miss.anova <- array(data=rep(0, length(resp.mer20.miss) \* 3), dim=c(length(resp.mer20.miss), 3))  
dimnames(mer.2020.sig.miss.anova)[[2]] <- c("treatment", "block", "interaction")  
dimnames(mer.2020.sig.miss.anova)[[1]] <- resp.mer20.miss  
  
mer.2020.sig.miss.art <- array(data=rep(0, length(resp.mer20.miss) \* 3), dim=c(length(resp.mer20.miss), 3))  
dimnames(mer.2020.sig.miss.art)[[2]] <- c("treatment", "block", "interaction")  
dimnames(mer.2020.sig.miss.art)[[1]] <- resp.mer20.miss  
  
mer.2020.sig.full.anova <- array(data=rep(0, length(resp.mer20.full) \* 3), dim=c(length(resp.mer20.full), 3))  
dimnames(mer.2020.sig.full.anova)[[2]] <- c("treatment", "block", "interaction")  
dimnames(mer.2020.sig.full.anova)[[1]] <- resp.mer20.full  
  
mer.2020.sig.full.art <- array(data=rep(0, length(resp.mer20.full) \* 3), dim=c(length(resp.mer20.full), 3))  
dimnames(mer.2020.sig.full.art)[[2]] <- c("treatment", "block", "interaction")  
dimnames(mer.2020.sig.full.art)[[1]] <- resp.mer20.full

resps = resp.mer20.full  
for (i in 1:length(resps)) {  
 print(resps[i])  
   
 res = aov( eval(parse(text=resps[i])) ~ treatment + block + treatment:block, merlot2020)  
 p.val = summary(res)[[1]][["Pr(>F)"]][c(1:3)]  
 mer.2020.sig.full.anova[i,] = p.val  
   
 print(summary(res))  
   
 model = art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, merlot2020)  
 res = anova(model, type = 1)  
 p.val = res[["Pr(>F)"]][c(1:3)]  
 mer.2020.sig.full.art[i,] = p.val  
   
 print(res)  
}

## [1] "fruitfulness"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.642 0.6421 2.251 0.140  
## block 5 2.569 0.5137 1.801 0.131  
## treatment:block 5 1.754 0.3508 1.230 0.310  
## Residuals 48 13.694 0.2853   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 3.2983 0.075600 .  
## 2 block 5 48 2.1377 0.076866 .  
## 3 treatment:block 5 48 1.7455 0.142324   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "SPAD"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 23.94 23.94 3.849 0.055598 .   
## block 5 190.74 38.15 6.133 0.000181 \*\*\*  
## treatment:block 5 18.94 3.79 0.609 0.693404   
## Residuals 48 298.58 6.22   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 4.21374 0.04557074 \*  
## 2 block 5 48 6.30709 0.00014203 \*\*\*  
## 3 treatment:block 5 48 0.65387 0.65998091   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "cluster.number"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 7.4 7.35 0.255 0.616  
## block 5 200.7 40.15 1.390 0.245  
## treatment:block 5 173.1 34.63 1.199 0.324  
## Residuals 48 1386.0 28.87   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.18587 0.66831   
## 2 block 5 48 1.09680 0.37449   
## 3 treatment:block 5 48 0.85233 0.51985   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "yield"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.08 0.0835 0.094 0.760  
## block 5 6.83 1.3668 1.540 0.195  
## treatment:block 5 6.75 1.3503 1.521 0.201  
## Residuals 48 42.60 0.8875   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.065643 0.79888   
## 2 block 5 48 1.052390 0.39834   
## 3 treatment:block 5 48 1.070013 0.38874   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "cluster.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)   
## treatment 1 0.00056 0.000562 0.161 0.6899   
## block 5 0.03873 0.007745 2.220 0.0675 .  
## treatment:block 5 0.02764 0.005529 1.585 0.1824   
## Residuals 48 0.16746 0.003489   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.15043 0.69984   
## 2 block 5 48 1.91794 0.10872   
## 3 treatment:block 5 48 1.60413 0.17704   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berries.cluster"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 544 544 0.260 0.612  
## block 5 17351 3470 1.659 0.163  
## treatment:block 5 15176 3035 1.451 0.224  
## Residuals 48 100420 2092   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.89051 0.35007   
## 2 block 5 48 1.76848 0.13733   
## 3 treatment:block 5 48 1.64398 0.16652   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
## [1] "berry.weight"  
## Df Sum Sq Mean Sq F value Pr(>F)  
## treatment 1 0.000 0.0000 0.000 1.000  
## block 5 1.682 0.3364 1.667 0.161  
## treatment:block 5 0.441 0.0882 0.437 0.820  
## Residuals 48 9.685 0.2018   
## Analysis of Variance of Aligned Rank Transformed Data  
##   
## Table Type: Analysis of Variance Table (Type I)   
## Model: No Repeated Measures (lm)  
## Response: art(eval(parse(text = resps[i])))  
##   
## Df Df.res F value Pr(>F)   
## 1 treatment 1 48 0.01190 0.91359   
## 2 block 5 48 1.38144 0.24797   
## 3 treatment:block 5 48 0.32561 0.89510   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

print(mer.2020.sig.full.anova)

## treatment block interaction  
## fruitfulness 0.14010568 0.1306129262 0.3098617  
## SPAD 0.05559777 0.0001808333 0.6934040  
## cluster.number 0.61620087 0.2446546446 0.3237969  
## yield 0.76041156 0.1952843379 0.2008933  
## cluster.weight 0.68988385 0.0674548550 0.1823613  
## berries.cluster 0.61230394 0.1627864349 0.2235099  
## berry.weight 1.00000000 0.1606009957 0.8203054

print(mer.2020.sig.full.art)

## treatment block interaction  
## fruitfulness 0.07560044 0.0768661137 0.1423241  
## SPAD 0.04557074 0.0001420307 0.6599809  
## cluster.number 0.66830893 0.3744901225 0.5198534  
## yield 0.79888488 0.3983404018 0.3887389  
## cluster.weight 0.69983513 0.1087182814 0.1770393  
## berries.cluster 0.35006515 0.1373335352 0.1665249  
## berry.weight 0.91358970 0.2479737695 0.8950960

mer.2020.sig.full = array(data=rep(0, length(resp.mer20.full) \* 2), dim=c(length(resp.mer20.full), 2))  
dimnames(mer.2020.sig.full)[[2]] <- c("treatment", "interaction")  
dimnames(mer.2020.sig.full)[[1]] <- resp.mer20.full  
for (i in 1:length(resp.mer20.full)) {  
 for (j in c(1,3)) {  
 if (j == 1) {  
 mer.2020.sig.full[i,j] = max(mer.2020.sig.full.anova[i,j], mer.2020.sig.full.art[i,j])  
 mer.2020.sig.full[i,j] = (mer.2020.sig.full[i,j] <= 0.05)   
 } else {  
 mer.2020.sig.full[i,j-1] = max(mer.2020.sig.full.anova[i,j], mer.2020.sig.full.art[i,j])  
 mer.2020.sig.full[i,j-1] = (mer.2020.sig.full[i,j-1] <= 0.05)   
 }  
 }  
}  
print(mer.2020.sig.full)

## treatment interaction  
## fruitfulness 0 0  
## SPAD 0 0  
## cluster.number 0 0  
## yield 0 0  
## cluster.weight 0 0  
## berries.cluster 0 0  
## berry.weight 0 0

resps = resp.mer20.miss  
for (i in 1:length(resps)) {  
 print(resps[i])  
   
 model\_string = paste(resps[i], "~ treatment + block + treatment:block", sep=" ")  
 res = mi.anova(merlot2020mice, model\_string, type = 3)  
 p.val = res$anova.table[["Pr(>F)"]][c(1:3)]  
 mer.2020.sig.miss.anova[i,] = p.val  
   
 print(res$anova.table)  
   
 d1 = complete(merlot2020mice, 1)  
 d2 = complete(merlot2020mice, 2)  
 d3 = complete(merlot2020mice, 3)  
 d4 = complete(merlot2020mice, 4)  
 d5 = complete(merlot2020mice, 5)  
   
 res1 = anova(art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, data = d1), type = 2)  
 res2 = anova(art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, data = d2), type = 2)  
 res3 = anova(art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, data = d3), type = 2)  
 res4 = anova(art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, data = d4), type = 2)  
 res5 = anova(art(eval(parse(text=resps[i])) ~ treatment + block + treatment:block, data = d5), type = 2)  
   
 for (j in 1:3){  
 fval = c(  
 res1[["F value"]][j],  
 res2[["F value"]][j],  
 res3[["F value"]][j],  
 res4[["F value"]][j],  
 res5[["F value"]][j]  
 )  
 comb.p = micombine.F(fval, res1[["Df"]][j])  
 mer.2020.sig.miss.art[i,j] = comb.p[["p"]]  
 }  
}

## [1] "berry.Brix"  
## Univariate ANOVA for Multiply Imputed Data (Type 3)   
##   
## lm Formula: berry.Brix ~ treatment + block + treatment:block  
## R^2=0.2643   
## ..........................................................................  
## ANOVA Table   
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 3.07495 1 341523.75 1.5960 0.20647 0.02455 0.03229  
## block 9.37886 5 6635180.01 0.9766 0.43032 0.07488 0.09238  
## treatment:block 20.65343 5 96415.05 2.1401 0.05767 0.16490 0.18311  
## Residual 92.14015 NA NA NA NA NA NA  
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 3.074952 1 341523.75 1.5960 0.206467 0.024551 0.032295  
## block 9.378856 5 6635180.01 0.9766 0.430320 0.074883 0.092385  
## treatment:block 20.653432 5 96415.05 2.1401 0.057667 0.164901 0.183108  
## Residual 92.140153 NA NA NA NA NA NA  
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(1, 361699.54)=1.787 p=0.18129   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 6617960.63)=1.066 p=0.37692   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 21523.47)=1.74 p=0.12169   
## [1] "berry.TA"  
## Univariate ANOVA for Multiply Imputed Data (Type 3)   
##   
## lm Formula: berry.TA ~ treatment + block + treatment:block  
## R^2=0.2441   
## ..........................................................................  
## ANOVA Table   
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 0.03007 1 12288160.3 0.0150 0.90246 0.00025 0.00033  
## block 16.60151 5 51863.3 1.7567 0.11804 0.13950 0.15581  
## treatment:block 12.42247 5 2794345.1 1.3245 0.25028 0.10439 0.12135  
## Residual 89.94907 NA NA NA NA NA NA  
## SSQ df1 df2 F value Pr(>F) eta2  
## treatment 0.0300654 1 12288160.3 0.0150 0.902456 0.000253  
## block 16.6015120 5 51863.3 1.7567 0.118038 0.139505  
## treatment:block 12.4224728 5 2794345.1 1.3245 0.250276 0.104388  
## Residual 89.9490742 NA NA NA NA NA  
## partial.eta2  
## treatment 0.000334  
## block 0.155809  
## treatment:block 0.121347  
## Residual NA  
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(1, 152415)=0.051 p=0.82088   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 250182.32)=1.92 p=0.08748   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 568739.93)=1.455 p=0.20082   
## [1] "berry.pH"  
## Univariate ANOVA for Multiply Imputed Data (Type 3)   
##   
## lm Formula: berry.pH ~ treatment + block + treatment:block  
## R^2=0.1153   
## ..........................................................................  
## ANOVA Table   
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 0.00073 1 255587066 0.0279 0.86744 0.00052 0.00058  
## block 0.08375 5 2321241 0.6403 0.66894 0.05914 0.06265  
## treatment:block 0.07880 5 189624669 0.6037 0.69713 0.05564 0.05917  
## Residual 1.25294 NA NA NA NA NA NA  
## SSQ df1 df2 F value Pr(>F) eta2  
## treatment 0.000733527 1 255587066 0.0279 0.867436 0.000518  
## block 0.083749679 5 2321241 0.6403 0.668937 0.059136  
## treatment:block 0.078801994 5 189624669 0.6037 0.697128 0.055642  
## Residual 1.252941757 NA NA NA NA NA  
## partial.eta2  
## treatment 0.000585  
## block 0.062654  
## treatment:block 0.059172  
## Residual NA  
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(1, 37768080.02)=0 p=0.98302   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 183586636.7)=0.895 p=0.48311   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 1372179.49)=0.627 p=0.67946   
## [1] "brown.seed.color"  
## Univariate ANOVA for Multiply Imputed Data (Type 3)   
##   
## lm Formula: brown.seed.color ~ treatment + block + treatment:block  
## R^2=0.2867   
## ..........................................................................  
## ANOVA Table   
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 4.95526 1 9505826.5 0.0717 0.78887 0.00109 0.00152  
## block 841.99272 5 32231829.5 2.4831 0.02952 0.18462 0.20560  
## treatment:block 460.39094 5 403905.5 1.3545 0.23810 0.10095 0.12397  
## Residual 3253.38559 NA NA NA NA NA NA  
## SSQ df1 df2 F value Pr(>F) eta2  
## treatment 4.955256 1 9505826.5 0.0717 0.788868 0.001087  
## block 841.992717 5 32231829.5 2.4831 0.029517 0.184618  
## treatment:block 460.390939 5 403905.5 1.3545 0.238104 0.100947  
## Residual 3253.385593 NA NA NA NA NA  
## partial.eta2  
## treatment 0.001521  
## block 0.205596  
## treatment:block 0.123968  
## Residual NA  
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(1, 8168037.28)=0.218 p=0.64093   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 10312461.05)=2.549 p=0.02591   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 1141461.68)=1.29 p=0.26471   
## [1] "pruning.weight"  
## Univariate ANOVA for Multiply Imputed Data (Type 3)   
##   
## lm Formula: pruning.weight ~ treatment + block + treatment:block  
## R^2=0.3028   
## ..........................................................................  
## ANOVA Table   
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 0.07462 1 2177.7361 3.2079 0.07342 0.04948 0.06628  
## block 0.16201 5 5590.3847 1.4291 0.21024 0.10744 0.13353  
## treatment:block 0.22003 5 614.4639 1.8308 0.10480 0.14591 0.17307  
## Residual 1.05130 NA NA NA NA NA NA  
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 0.07461959 1 2177.7361 3.2079 0.073423 0.049484 0.066275  
## block 0.16200767 5 5590.3847 1.4291 0.210237 0.107435 0.133526  
## treatment:block 0.22003043 5 614.4639 1.8308 0.104802 0.145913 0.173071  
## Residual 1.05129684 NA NA NA NA NA NA  
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(1, 639.85)=2.727 p=0.09915   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 1904.91)=1.23 p=0.29236   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 506.42)=2.282 p=0.04545   
## [1] "Ravaz.index"  
## Univariate ANOVA for Multiply Imputed Data (Type 3)   
##   
## lm Formula: Ravaz.index ~ treatment + block + treatment:block  
## R^2=0.406   
## ..........................................................................  
## ANOVA Table   
## SSQ df1 df2 F value Pr(>F) eta2 partial.eta2  
## treatment 0.87958 1 5532.65077 0.0539 0.81634 0.00120 0.00202  
## block 284.53954 5 36.23581 4.6771 0.00214 0.38838 0.39535  
## treatment:block 12.03771 5 10331.02017 0.2447 0.94259 0.01643 0.02692  
## Residual 435.17550 NA NA NA NA NA NA  
## SSQ df1 df2 F value Pr(>F) eta2  
## treatment 0.879576 1 5532.65077 0.0539 0.816345 0.001201  
## block 284.539540 5 36.23581 4.6771 0.002141 0.388380  
## treatment:block 12.037714 5 10331.02017 0.2447 0.942588 0.016431  
## Residual 435.175497 NA NA NA NA NA  
## partial.eta2  
## treatment 0.002017  
## block 0.395350  
## treatment:block 0.026917  
## Residual NA  
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(1, 12142.22)=0.007 p=0.93177   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 26.86)=3.66 p=0.0118   
## Combination of Chi Square Statistics for Multiply Imputed Data  
## Using 5 Imputed Data Sets  
## F(5, 6221.17)=0.411 p=0.84153

print(mer.2020.sig.miss.anova)

## treatment block interaction  
## berry.Brix 0.206467 0.430320 0.057667  
## berry.TA 0.902456 0.118038 0.250276  
## berry.pH 0.867436 0.668937 0.697128  
## brown.seed.color 0.788868 0.029517 0.238104  
## pruning.weight 0.073423 0.210237 0.104802  
## Ravaz.index 0.816345 0.002141 0.942588

print(mer.2020.sig.miss.art)

## treatment block interaction  
## berry.Brix 0.18128987 0.37692439 0.1216890  
## berry.TA 0.82087829 0.08747579 0.2008182  
## berry.pH 0.98302494 0.48311103 0.6794568  
## brown.seed.color 0.64093323 0.02590514 0.2647140  
## pruning.weight 0.09915411 0.29236034 0.0454518  
## Ravaz.index 0.93177493 0.01179558 0.8415292

mer.2020.sig.miss = array(data=rep(0, length(resp.mer20.miss) \* 2), dim=c(length(resp.mer20.miss), 2))  
dimnames(mer.2020.sig.miss)[[2]] <- c("treatment", "interaction")  
dimnames(mer.2020.sig.miss)[[1]] <- resp.mer20.miss  
for (i in 1:length(resp.mer20.miss)) {  
 for (j in c(1,3)) {  
 if (j == 1) {  
 mer.2020.sig.miss[i,j] = max(mer.2020.sig.miss.anova[i,j], mer.2020.sig.miss.art[i,j])  
 mer.2020.sig.miss[i,j] = (mer.2020.sig.miss[i,j] <= 0.05)   
 } else {  
 mer.2020.sig.miss[i,j-1] = max(mer.2020.sig.miss.anova[i,j], mer.2020.sig.miss.art[i,j])  
 mer.2020.sig.miss[i,j-1] = (mer.2020.sig.miss[i,j-1] <= 0.05)   
 }  
 }  
}  
print(mer.2020.sig.miss)

## treatment interaction  
## berry.Brix 0 0  
## berry.TA 0 0  
## berry.pH 0 0  
## brown.seed.color 0 0  
## pruning.weight 0 0  
## Ravaz.index 0 0

# Subgroup analysis and estimated effects

# get names of responses that we need to do subgroup analysis on (interaction)  
which(mer.2019.sig[,2] == 1)

## berry.weight berry.pH pruning.weight Ravaz.index   
## 6 8 10 11

sub\_response = resp.mer19[which(mer.2019.sig[,2] == 1)]  
  
for (i in 1:length(sub\_response)) {  
 print(sub\_response[i])  
 res = aov( eval(parse(text=sub\_response[i])) ~ treatment + block + treatment:block, merlot2019)  
 print(tukey\_hsd(res, which = "treatment:block"))  
}

## [1] "berry.weight"  
## # A tibble: 66 x 9  
## term group1 group2 null.value estimate conf.low conf.high p.adj p.adj.signif  
## \* <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   
## 1 trea~ contr~ heat:1 0 0.133 -0.170 0.437 0.931 ns   
## 2 trea~ contr~ contr~ 0 0.0187 -0.285 0.322 1 ns   
## 3 trea~ contr~ heat:2 0 0.0911 -0.213 0.395 0.996 ns   
## 4 trea~ contr~ contr~ 0 0.00593 -0.298 0.310 1 ns   
## 5 trea~ contr~ heat:3 0 0.0611 -0.243 0.365 1 ns   
## 6 trea~ contr~ contr~ 0 0.103 -0.201 0.407 0.989 ns   
## 7 trea~ contr~ heat:4 0 -0.0612 -0.365 0.243 1 ns   
## 8 trea~ contr~ contr~ 0 -0.108 -0.412 0.196 0.985 ns   
## 9 trea~ contr~ heat:5 0 -0.243 -0.546 0.0610 0.237 ns   
## 10 trea~ contr~ contr~ 0 -0.210 -0.513 0.0940 0.444 ns   
## # ... with 56 more rows  
## [1] "berry.pH"  
## # A tibble: 66 x 9  
## term group1 group2 null.value estimate conf.low conf.high p.adj p.adj.signif  
## \* <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   
## 1 trea~ contr~ heat:1 0 -0.108 -0.256 0.0404 0.366 ns   
## 2 trea~ contr~ contr~ 0 -0.126 -0.274 0.0224 0.168 ns   
## 3 trea~ contr~ heat:2 0 -0.11 -0.258 0.0384 0.339 ns   
## 4 trea~ contr~ contr~ 0 -0.07 -0.218 0.0784 0.893 ns   
## 5 trea~ contr~ heat:3 0 -0.046 -0.194 0.102 0.995 ns   
## 6 trea~ contr~ contr~ 0 -0.134 -0.282 0.0144 0.112 ns   
## 7 trea~ contr~ heat:4 0 -0.04 -0.188 0.108 0.998 ns   
## 8 trea~ contr~ contr~ 0 -0.02 -0.168 0.128 1 ns   
## 9 trea~ contr~ heat:5 0 0.104 -0.0444 0.252 0.422 ns   
## 10 trea~ contr~ contr~ 0 -0.106 -0.254 0.0424 0.393 ns   
## # ... with 56 more rows  
## [1] "pruning.weight"  
## # A tibble: 66 x 9  
## term group1 group2 null.value estimate conf.low conf.high p.adj  
## \* <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 trea~ contr~ heat:1 0 -0.134 -0.446 0.178 0.941   
## 2 trea~ contr~ contr~ 0 -0.0454 -0.357 0.267 1   
## 3 trea~ contr~ heat:2 0 -0.245 -0.557 0.0669 0.258   
## 4 trea~ contr~ contr~ 0 -0.256 -0.568 0.0559 0.205   
## 5 trea~ contr~ heat:3 0 -0.262 -0.574 0.0497 0.178   
## 6 trea~ contr~ contr~ 0 -0.254 -0.566 0.0583 0.216   
## 7 trea~ contr~ heat:4 0 -0.245 -0.557 0.0669 0.258   
## 8 trea~ contr~ contr~ 0 -0.232 -0.544 0.0801 0.334   
## 9 trea~ contr~ heat:5 0 -0.193 -0.505 0.119 0.612   
## 10 trea~ contr~ contr~ 0 -0.333 -0.645 -0.0209 0.0272  
## # ... with 56 more rows, and 1 more variable: p.adj.signif <chr>  
## [1] "Ravaz.index"  
## # A tibble: 66 x 9  
## term group1 group2 null.value estimate conf.low conf.high p.adj p.adj.signif  
## \* <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   
## 1 trea~ contr~ heat:1 0 -0.683 -13.8 12.4 1 ns   
## 2 trea~ contr~ contr~ 0 -1.29 -14.4 11.8 1 ns   
## 3 trea~ contr~ heat:2 0 4.97 -8.13 18.1 0.975 ns   
## 4 trea~ contr~ contr~ 0 2.58 -10.5 15.7 1 ns   
## 5 trea~ contr~ heat:3 0 7.62 -5.48 20.7 0.693 ns   
## 6 trea~ contr~ contr~ 0 5.55 -7.54 18.7 0.945 ns   
## 7 trea~ contr~ heat:4 0 -0.499 -13.6 12.6 1 ns   
## 8 trea~ contr~ contr~ 0 4.94 -8.16 18.0 0.976 ns   
## 9 trea~ contr~ heat:5 0 -4.53 -17.6 8.57 0.988 ns   
## 10 trea~ contr~ contr~ 0 7.36 -5.74 20.5 0.735 ns   
## # ... with 56 more rows

# Repeated measures ANOVA

# Power calculation

# helper function to find the min difference in block levels  
min.da = function(data, resp){  
 meanresp<-data %>%   
 select(block, resp) %>%   
 group\_by(block) %>%  
 mutate(mean = mean(.data[[resp]])) %>%   
 select(block, mean) %>%   
 distinct()  
   
 sorted<-sort(meanresp$mean)  
 da=10000  
   
 for(i in 1:(length(sorted)-1)){  
 curdiff = sorted[i+1]-sorted[i]  
 if (curdiff < da){  
 da = curdiff  
 }else{  
 da = da  
 }  
 }  
 return(da)  
}  
  
# helper function find the min difference for treatment levels  
min.db = function(data,resp){  
 return(abs(mean(data[[resp]][which(data[["treatment"]]=="heat")])  
 -mean(data[[resp]][which(data[["treatment"]]=="control")])))  
}  
  
# function to calculate power  
power=function(data, resp){  
 pwr.2way(a=length(unique(data[["block"]])),   
 b=length(unique(data[["treatment"]])),   
 alpha=0.05,   
 size.A=nrow(data %>% filter(block == 1)),   
 size.B=nrow(data %>% filter(treatment=="heat")),  
 f.A=min.da(data, resp)/sd(data[[resp]]),   
 f.B=min.db(data, resp)/sd(data[[resp]]))  
}

## Chardonnay 2019

for (i in colnames(chardonnay2019[-1:-5])){  
 print(i)  
 print(power(chardonnay2019, i))  
}

## [1] "X..of.clusters"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 5  
## b = 2  
## n.A = 8  
## n.B = 20  
## sig.level = 0.05  
## power.A = 0.1105931  
## power.B = 0.9950189  
## power = 0.1105931  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "yield..kg."  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 5  
## b = 2  
## n.A = 8  
## n.B = 20  
## sig.level = 0.05  
## power.A = 0.05083315  
## power.B = 0.8112166  
## power = 0.05083315  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "av.cluster.weight..kg."  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 5  
## b = 2  
## n.A = 8  
## n.B = 20  
## sig.level = 0.05  
## power.A = 0.08182039  
## power.B = 0.05690024  
## power = 0.05690024  
##   
## NOTE: power is the minimum power among two factors

## Chardonnay 2020

for (i in colnames(chardonnay2020[-1:-5])){  
 print(i)  
 print(power(chardonnay2020, i))  
}

## [1] "X..of.clusters"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 5  
## b = 2  
## n.A = 8  
## n.B = 20  
## sig.level = 0.05  
## power.A = 0.05290326  
## power.B = 0.989015  
## power = 0.05290326  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "yield..kg."  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 5  
## b = 2  
## n.A = 8  
## n.B = 20  
## sig.level = 0.05  
## power.A = 0.09102965  
## power.B = 0.3080069  
## power = 0.09102965  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "av.cluster.weight..kg."  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 5  
## b = 2  
## n.A = 8  
## n.B = 20  
## sig.level = 0.05  
## power.A = 0.2167063  
## power.B = 0.9748234  
## power = 0.2167063  
##   
## NOTE: power is the minimum power among two factors

## Merlot 2019

for (i in colnames(merlot2019[c(-1:-5,-7)])){ # excluded veraison   
 print(i)  
 print(power(merlot2019, i))  
}

## [1] "SPAD"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.06195914  
## power.B = 0.9971874  
## power = 0.06195914  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "cluster.number"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.07967274  
## power.B = 0.9998222  
## power = 0.07967274  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "yield"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.4652131  
## power.B = 1  
## power = 0.4652131  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "cluster.weight"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05069867  
## power.B = 0.9980656  
## power = 0.05069867  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berries.cluster"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.0521321  
## power.B = 1  
## power = 0.0521321  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.weight"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.07360631  
## power.B = 0.9992952  
## power = 0.07360631  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.TA"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05333966  
## power.B = 0.8879783  
## power = 0.05333966  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.pH"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05061232  
## power.B = 0.9999978  
## power = 0.05061232  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.Brix"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05874802  
## power.B = 0.9998161  
## power = 0.05874802  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "pruning.weight"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05147242  
## power.B = 0.1936798  
## power = 0.05147242  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "Ravaz.index"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.0604988  
## power.B = 0.9999999  
## power = 0.0604988  
##   
## NOTE: power is the minimum power among two factors

## Merlot 2020

mer2020mice1<-complete(merlot2020mice,1)  
for (i in colnames(mer2020mice1[-1:-5])){  
 print(i)  
 print(power(mer2020mice1, i))  
}

## [1] "fruitfulness"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.1149216  
## power.B = 0.9999997  
## power = 0.1149216  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "bloom"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05372978  
## power.B = 0.9307511  
## power = 0.05372978  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "SPAD"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.0535127  
## power.B = 1  
## power = 0.0535127  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "veraison"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.1936121  
## power.B = 0.9906151  
## power = 0.1936121  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "cluster.number"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.07836625  
## power.B = 0.6773147  
## power = 0.07836625  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "yield"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.06354236  
## power.B = 0.3037632  
## power = 0.06354236  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "cluster.weight"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.06493687  
## power.B = 0.4514977  
## power = 0.06493687  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berries.cluster"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05263426  
## power.B = 0.6687976  
## power = 0.05263426  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.weight"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05441044  
## power.B = 0.05  
## power = 0.05  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.Brix"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05978902  
## power.B = 0.999991  
## power = 0.05978902  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.TA"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.08491847  
## power.B = 0.08334365  
## power = 0.08334365  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "berry.pH"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05522094  
## power.B = 0.1343025  
## power = 0.05522094  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "brown.seed.color"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.06438124  
## power.B = 0.2819041  
## power = 0.06438124  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "pruning.weight"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.05000014  
## power.B = 1  
## power = 0.05000014  
##   
## NOTE: power is the minimum power among two factors  
##   
## [1] "Ravaz.index"  
##   
## Balanced two-way analysis of variance power calculation   
##   
## a = 6  
## b = 2  
## n.A = 10  
## n.B = 30  
## sig.level = 0.05  
## power.A = 0.1097695  
## power.B = 0.1766168  
## power = 0.1097695  
##   
## NOTE: power is the minimum power among two factors