PASS1A DMAQC data: analysis by BIC

```
# Set the working directory to the folder with the data
dmaqc_data_dir = "/Users/David/Desktop/MoTrPAC/data/pass_1a/dmaqc_pheno/"
all_csvs = list.files(dmaqc_data_dir,full.names = T) # get all files in dir
all_csvs = all_csvs[grepl(".csv$",all_csvs)] # make sure we take csv only
# read all files
csv_data = list()
for(fname in all_csvs){
  csv_data[[fname]] = read.csv(fname,stringsAsFactors = F)
}# sapply(csv_data,dim) # check the dimensions of the different datasets

# dictionary path
dmaqc_dict_dir = "/Users/David/Desktop/MoTrPAC/data/pass_1a/dmaqc_pheno/dictionary/"
all_dict_csvs = list.files(dmaqc_dict_dir,full.names = T) # get all files in dir
all_dict_csvs = all_dict_csvs[grepl(".csv$",all_dict_csvs)] # make sure we take csv only
# read all files
dict_data = list()
for(fname in all_dict_csvs){
  dict_data[[fname]] = read.csv(fname,stringsAsFactors = F)
}
#sapply(dict_data,dim)
```

1 Sanity check: Acute tests basic statistics

```
# Get the acute test data
ac_test_data = csv_data[[which(grepl("Acute.Test",names(csv_data)))]]
dim(ac_test_data)
```

```
## [1] 108  23
```

```
# check the time differences between start and end
test_times = as.difftime(ac_test_data$t_complete) - as.difftime(ac_test_data$t_start)
# table of the values: all except for on are 0.5 hours
table(test_times)
```

```
## test_times
## 0.466666666666667               0.5
##                 1               107
```

```
# Get the comment of the sample that is not 0.5h
ac_test_data[test_times!=0.5,"comments"]
```

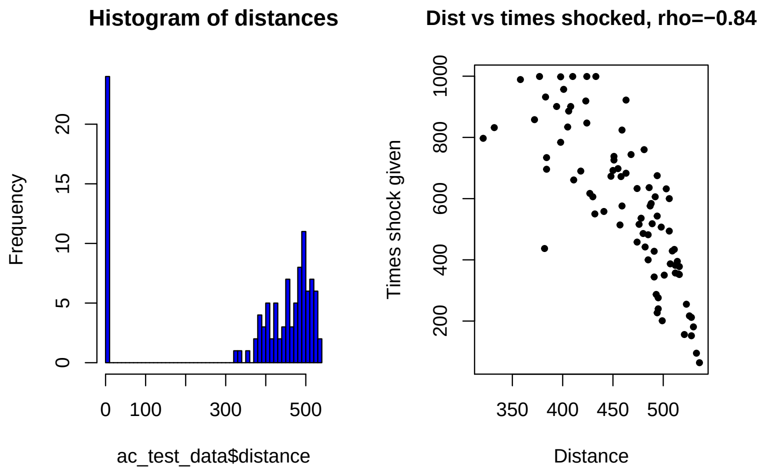
```
## [1] "Treadmill stopped 28:49 (mm:ss) into the acute bout due to problems with the other rat on the s
```

```
ac_test_data$formatted_test_time = test_times
```

Next, we analyze the distances. We illustrate how these are a function of the shocks and sex/weight.

```
# convert the shock lengths to numbers (seconds)
parse_shocktime<-function(x){
  arr = strsplit(x,split=":")[[1]]
  if(length(arr)<2){return(NA)}
```

```r
  return(as.numeric(arr[1])*60+as.numeric(arr[2]))
}
tmp_x = ac_test_data$howlongshock
tmp_x = sapply(tmp_x, parse_shocktime)
ac_test_data$howlongshock = tmp_x
rm(tmp_x)

par(mfrow=c(1,2))
# histogram of distances
hist(ac_test_data$distance,col="blue",breaks=50,main = "Histogram of distances")

# Correlation between distance and number of shocks
# Get the indices of the samples with shock information -
# these the animals that did the acute test
timesshock_inds = !is.na(ac_test_data$timesshock)
# create a new dataframe with the selected animals
trained_animals_data = ac_test_data[timesshock_inds,]
sp_corr = cor(trained_animals_data$distance,
              trained_animals_data$timesshock,method="spearman")
plot(trained_animals_data$distance,trained_animals_data$timesshock,
     main=paste("Dist vs times shocked, rho=",format(sp_corr,digits = 2),sep=""),
     pch=20,ylab="Times shock given",xlab="Distance",cex.main=1.1)
```



```r
# A "smarter" analysis: regression of the distance using shock info
dist_lm  = lm(distance~timesshock+howlongshock+weight+days_start,
              data=trained_animals_data)
# Summary of the model, points to take: high R^2, significance of
# the features
summary(dist_lm)
```

```
##
## Call:
## lm(formula = distance ~ timesshock + howlongshock + weight +
##     days_start, data = trained_animals_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.1248  -1.0430   0.6867   2.4416   8.8814
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  562.127804   2.427681 231.549   <2e-16 ***
## timesshock     0.003171   0.003464   0.916    0.363
## howlongshock  -0.296415   0.005700 -52.004   <2e-16 ***
## weight        -0.147827   0.006563 -22.526   <2e-16 ***
## days_start     0.032731   0.024534   1.334    0.186
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.623 on 79 degrees of freedom
## Multiple R-squared:  0.9921, Adjusted R-squared:  0.9917
## F-statistic:  2466 on 4 and 79 DF,  p-value: < 2.2e-16
```

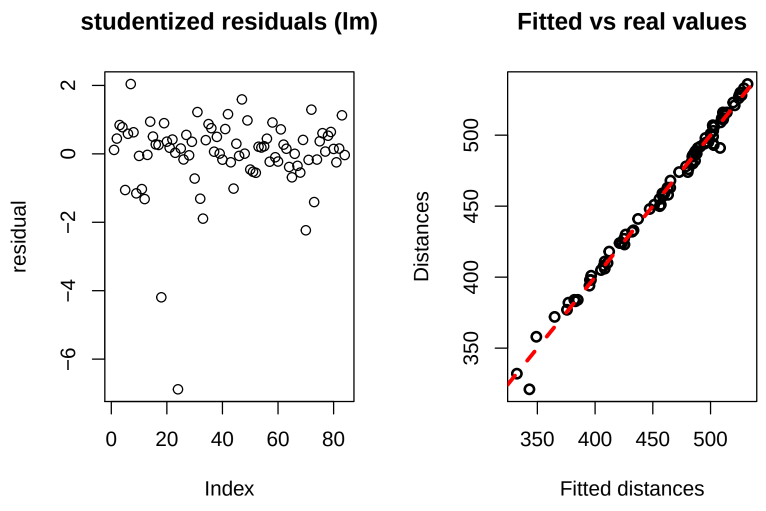
```
# We have some clear outliers:
library(MASS)
par(mfrow=c(1,2))
plot(studres(dist_lm),main="studentized residuals (lm)",ylab="residual")
# Select the top outliers and look at their comments
outliers = abs(studres(dist_lm)) > 2
# how many outliers have we selected?
sum(outliers)
```

```
## [1] 4
```

```
# their comments:
trained_animals_data[outliers,"comments"]
```

```
## [1] "Increased shock at 20 min."
## [2] "Treadmill stopped 28:49 (mm:ss) into the acute bout due to problems with the other rat on the s
## [3] "Shock grid increased to 1.0 mA at 22 minutes. Treadmill bout stopped at 28:49 (mm:ss) due to an
## [4] ""
```

```
# Plot the fitted values of the linear regression vs.
# the true distances
plot(dist_lm$fitted.values,trained_animals_data$distance,lwd=2,
     main="Fitted vs real values",ylab="Distances",xlab="Fitted distances")
abline(0,1,col="red",lty=2,lwd=3)
```



2 Site comparison

In some versions of the DMAQC data there is a single site. In this case this section will not result in an output.

```
# Load additional information about the animals
registr_data = csv_data[[which(grepl("Regist",names(csv_data)))]]
rownames(registr_data) = as.character(registr_data$pid)
# make the rownames in the test data comparable
rownames(trained_animals_data) = trained_animals_data$pid
# add sex to the trained animal data data frame
sex_key = c("Female","Male")
trained_animals_data$sex = sex_key[registr_data[rownames(trained_animals_data),"sex"]]

# Map site Ids to their names
site_names = c("910"="Joslin","930"="Florida")
trained_animals_data$site = site_names[as.character(trained_animals_data$siteID)]

# Sanity check: the numbers should be the same for both sites
table(ac_test_data$siteID)
```

```
##
## 910
## 108
```

```
table(trained_animals_data$site,trained_animals_data$sex)
```

```
##
##          Female Male
##   Joslin     42   42
```

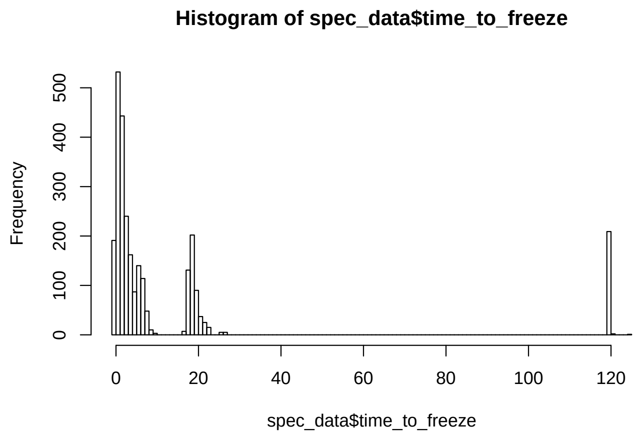
```
run_wilcox<-function(x1,x2){
  return(wilcox.test(x1[x2==x2[1]],x1[x2!=x2[1]])$p.value)
}
# Compare the distances, shocks, and weight (if we have multiple site)
if (length(unique(ac_test_data$siteID))>1){
  par(mfrow=c(1,3),mar=c(10,4,4,4))
  # Site only
  p_dist = run_wilcox(trained_animals_data$distance,trained_animals_data$site)
  boxplot(distance~site,data=trained_animals_data,col="cyan",ylab="Distance",
        main=paste("Site vs. distance, p<",format(p_dist,digits = 2)),
        cex.main=1,las=2)
  p_timesshock = run_wilcox(trained_animals_data$timesshock,trained_animals_data$site)
  boxplot(timesshock~site,data=trained_animals_data,col="red",ylab="Times shocked",
        main=paste("Site vs. times shocked, p<",format(p_timesshock,digits = 3)),
        cex.main=1,las=2)
  p_w = run_wilcox(trained_animals_data$weight,trained_animals_data$site)
  boxplot(weight~site,data=trained_animals_data,col="cyan",ylab="Weight",
        main=paste("Site vs. weight, p=",format(p_w,digits = 2)),
        cex.main=1,las=2)
  # Site and sex
  par(mfrow=c(1,3),mar=c(10,4,4,4))
  boxplot(distance~site+sex,data=trained_animals_data,col="cyan",ylab="Distance",
        main="Site vs. distance",cex.main=1,las=2)
  boxplot(timesshock~site+sex,data=trained_animals_data,col="red",ylab="Times shocked",
        main="Site vs. times shocked",cex.main=1,las=2)
boxplot(weight~site+sex,data=trained_animals_data,col="cyan",ylab="Weight",
      main="Site vs. weight",cex.main=1,las=2)

# Regress time shocked and distance vs. site and sex
summary(lm(timesshock~site+sex,data=trained_animals_data))
summary(lm(distance~site+sex,data=trained_animals_data))
}
```

3 Sanity checks: Biospecimen data

```
# Analysis of biospecimen data
spec_data = csv_data[[which(grepl("Specimen.Processing.csv",names(csv_data)))]]
rownames(spec_data) = spec_data$labelid
# Parse the times and compute the difference between the freeze time and
# the collection time
time_to_freeze1 = as.difftime(spec_data$t_freeze,units = "mins") -
  as.difftime(spec_data$t_collection,units="mins")
# For some samples we have the edta spin time instead of the collection
# time, use these when there are no other options
time_to_freeze2 = as.difftime(spec_data$t_freeze,units = "mins") -
  as.difftime(spec_data$t_edtaspin,units="mins")
time_to_freeze = time_to_freeze1
# Fill in the NAs by taking the time between the edta spin and the freeze
table(is.na(time_to_freeze1),is.na(time_to_freeze2))
```

```
##
##         FALSE TRUE
##   FALSE     0 2182
##   TRUE    517    0
```

```
time_to_freeze[is.na(time_to_freeze1)] = time_to_freeze2[is.na(time_to_freeze1)]
spec_data$time_to_freeze = as.numeric(time_to_freeze)
spec_data$time_to_freeze_from_collection = as.numeric(time_to_freeze1)
spec_data$time_to_freeze_from_edta_spin = as.numeric(time_to_freeze2)
hist(spec_data$time_to_freeze,breaks = 100)
```

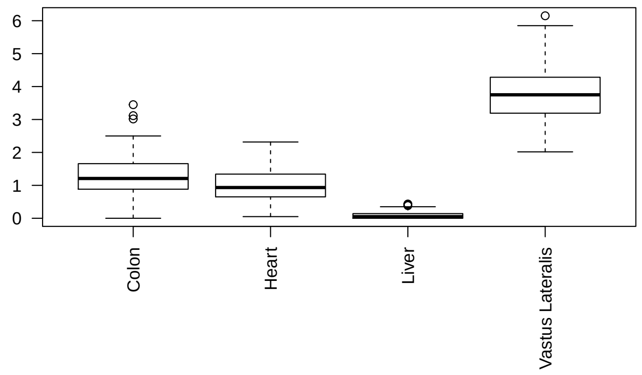


```
# Add site by name
site_names = c("910"="Joslin","930"="Florida")
```

```
spec_data$site = site_names[as.character(spec_data$siteid)]
table(spec_data$site)
```

```
##
## Joslin
##   2699
```

```
inds = !is.na(time_to_freeze1)
inds = grepl("adipose",spec_data$sampletypedescription,ignore.case = T)
inds = grepl("heart",spec_data$sampletypedescription,ignore.case = T) |
  grepl("liver",spec_data$sampletypedescription,ignore.case = T) |
  grepl("colon",spec_data$sampletypedescription,ignore.case = T) |
  grepl("vastus",spec_data$sampletypedescription,ignore.case = T)
# Using site info:
# Here we use an interaction term and not addition as the R^2 is >2 times
# greater this way
if (length(unique(spec_data$site))>1){
  par(mar=c(10,2,2,2))
  boxplot(time_to_freeze~site:sampletypedescription,data=spec_data[inds,],
        ylab="Time to freeze",las=2)
  summary(lm(time_to_freeze~sampletypedescription:site,data=spec_data[inds,]))
}
# A single site
if (length(unique(spec_data$site))==1){
  par(mar=c(10,2,2,2))
  boxplot(time_to_freeze~sampletypedescription,data=spec_data[inds,],
        ylab="Time to freeze",las=2)
  summary(lm(time_to_freeze~sampletypedescription,data=spec_data[inds,]))
}
```



```
##
## Call:
## lm(formula = time_to_freeze ~ sampletypedescription, data = spec_data[inds,
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.78765 -0.33731 -0.05216  0.27994  2.34568
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                              1.32052    0.05478  24.106  < 2e-16
## sampletypedescriptionHeart              -0.30046    0.07747  -3.878 0.000122
## sampletypedescriptionLiver           -1.23503    0.07747 -15.942  < 2e-16
## sampletypedescriptionVastus Lateralis  2.48380    0.07747  32.061  < 2e-16
##
## (Intercept)                           ***
## sampletypedescriptionHeart            ***
## sampletypedescriptionLiver            ***
## sampletypedescriptionVastus Lateralis ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5693 on 428 degrees of freedom
## Multiple R-squared:  0.8548, Adjusted R-squared:  0.8538
## F-statistic: 839.8 on 3 and 428 DF,  p-value: < 2.2e-16
```

4 Format the metadata table according to vial ids

We now use DMAQC's mapping of label ids to vial ids and use it to generate a single metadata table that we can share with other sites.

```r
# Helper function for merging columns from data2 into data1
# The function makes sure there is no column duplications when
# adding information from data2 into data1
merge_avoid_col_dup<-function(data1,data2,by_col){
  data2_cols = c(by_col,setdiff(colnames(data2),colnames(data1)))
  res = merge(data1, data2[,data2_cols], by=by_col)
  return(res)
}
# Note that Specimen.Processing is intentionally the last added dataset
# We merge by PIDs so all data before that are animal-level data
formnames = c("Acute.Test","Animal.Familiarization",
              "Animal.Key","Animal.Registration",
              "Specimen.Collection","Specimen.Processing")
merged_dmaqc_data = c()
for(currname in formnames){
  curr_data = csv_data[[which(grepl(currname,names(csv_data)))]]
  colnames(curr_data) = paste(currname,colnames(curr_data),sep=".")
  colnames(curr_data)[grepl(".pid$",colnames(curr_data))]="pid"
  colnames(curr_data)[grepl(".bid$",colnames(curr_data))]="bid"
  colnames(curr_data)[grepl(".vialid$",colnames(curr_data))]="vialid"
  colnames(curr_data)[grepl(".viallabel$",colnames(curr_data))]="viallabel"
  colnames(curr_data)[grepl(".labelid$",colnames(curr_data))]="labelid"
  colnames(curr_data) = tolower(colnames(curr_data))
  by_col = "pid"
  if(length(merged_dmaqc_data)==0){
    merged_dmaqc_data = curr_data
  }
  else{
    merged_dmaqc_data = merge_avoid_col_dup(merged_dmaqc_data,curr_data,by_col)
  }
}
print("Merged animal and biospecimen data tables, dim is:")
```

```
## [1] "Merged animal and biospecimen data tables, dim is:"
```

```
print(dim(merged_dmaqc_data))
```

```
## [1] 2699  106
```

```
# Now map DMAQC's label ids to vialids
# Sort to make the most up to date file the first in the order
mapping_files = sort(all_csvs[grepl("BICLabelData",all_csvs)],decreasing = T)
mapping_info = csv_data[[mapping_files[1]]]
colnames(mapping_info) = tolower(colnames(mapping_info))
# Not all samples in the specimen data are necessarily covered in the mapping
# file. The mapping file contains info only about samples that were shipped
# to CAS. As can be seen here:
table(is.element(merged_dmaqc_data$labelid,set=mapping_info$labelid))
```

```
##
## FALSE  TRUE
##  1061  1638
```

```
# We therefore need to extract the intersection:
shared_labelids = intersect(merged_dmaqc_data$labelid,mapping_info$labelid)
merged_dmaqc_data = merged_dmaqc_data[
  is.element(merged_dmaqc_data$labelid,set = shared_labelids),]
mapping_info = mapping_info[
  is.element(mapping_info$labelid,set = shared_labelids),]
print("Merged animal and biospecimen data tables, new dim is:")
```

```
## [1] "Merged animal and biospecimen data tables, new dim is:"
```

```
print(dim(merged_dmaqc_data))
```

```
## [1] 1638  106
```

```
# We also have a many to one mapping from vial ids to labels, we
# merge the tables to avoid information loss
merged_dmaqc_data = merge_avoid_col_dup(merged_dmaqc_data,mapping_info,"labelid")
print("Merged animal and biospecimen data tables, after adding vialids, new dim is:")
```

```
## [1] "Merged animal and biospecimen data tables, after adding vialids, new dim is:"
```

```
print(dim(merged_dmaqc_data))
```

```
## [1] 8616  109
```

```
# Now put the dictionary in one file as well
merged_column_dictionary = c()
cols_to_take = c("Field.Name","Data.Type","Categorical.Values",
                 "Categorical.Definitions")
for(currname in formnames){
  tmp_dict_data = dict_data[[which(grepl(currname,names(dict_data)))]]
  tmp_dict_data = tmp_dict_data[,cols_to_take]
  tmp_dict_data[,1] = paste(currname,tmp_dict_data[,1],sep=".")
  tmp_dict_data[grepl(".pid$",tmp_dict_data[,1]),1]="pid"
  tmp_dict_data[grepl(".bid$",tmp_dict_data[,1]),1]="bid"
  tmp_dict_data[grepl(".labelid$",tmp_dict_data[,1]),1]="labelid"
  tmp_dict_data[grepl(".vialid$",tmp_dict_data[,1]),1]="vialid"
  tmp_dict_data[grepl(".viallabel$",tmp_dict_data[,1]),1]="viallabel"
  tmp_dict_data[,1] = tolower(tmp_dict_data[,1])
  tmp_dict_data = cbind(tmp_dict_data,rep(currname,nrow(tmp_dict_data)))
  merged_column_dictionary = rbind(merged_column_dictionary,tmp_dict_data)
}

# Add the calculated features
formnames = c(formnames,"Calculated.Variables")
currname = "Calculated.Variables"
# Data
curr_data = csv_data[[which(grepl(currname,names(csv_data)))]]
colnames(curr_data) = paste(currname,colnames(curr_data),sep=".")
colnames(curr_data)[grepl(".labelid$",colnames(curr_data))]="labelid"
colnames(curr_data) = tolower(colnames(curr_data))
merged_dmaqc_data = merge_avoid_col_dup(merged_dmaqc_data,curr_data,"labelid")
# Dictionary
tmp_dict_data = dict_data[[which(grepl(currname,names(dict_data)))]]
tmp_dict_data = tmp_dict_data[,cols_to_take]
tmp_dict_data[,1] = paste(currname,tmp_dict_data[,1],sep=".")
tmp_dict_data[grepl(".pid$",tmp_dict_data[,1]),1]="pid"
tmp_dict_data[grepl(".bid$",tmp_dict_data[,1]),1]="bid"
tmp_dict_data[grepl(".labelid$",tmp_dict_data[,1]),1]="labelid"
tmp_dict_data[,1] = tolower(tmp_dict_data[,1])
tmp_dict_data = cbind(tmp_dict_data,rep(currname,nrow(tmp_dict_data)))
merged_column_dictionary = rbind(merged_column_dictionary,tmp_dict_data)

# Final checks of the data
dim(merged_dmaqc_data)
```

```
## [1] 8616  116
```

```
merged_column_dictionary = merged_column_dictionary[is.element(
  merged_column_dictionary[,1],set=colnames(merged_dmaqc_data)
  ),]
dim(merged_column_dictionary)
```

```
## [1] 155   5
```

```
merged_column_dictionary = unique(merged_column_dictionary)
```

5 Compare to the DMAQC computed scores

As requested by Ashley (email from June 7 2019), the following computed fields were added by DMQAQC:

1. Weight gain before acute test: (Animal_Acute_Test.weight – Animal_Registration.weight)
2. Lactate changes due to acute exercise: (Aminal_Acute_Test.endblood - Aminal_Acute_Test.beginblood)
3. EDTA sample collection time: (Animal_Specimen_Collection.t_edtafill - Aminal_Acute_Test.t_complete)
4. Time of death after acute test: (Animal_Specimen_Collection.t_death - Aminal_Acute_Test.t_complete)
5. Sample frozen time after acute test: (Animal_Sample_Processing.t_freeze - Aminal_Acute_Test.t_complete)

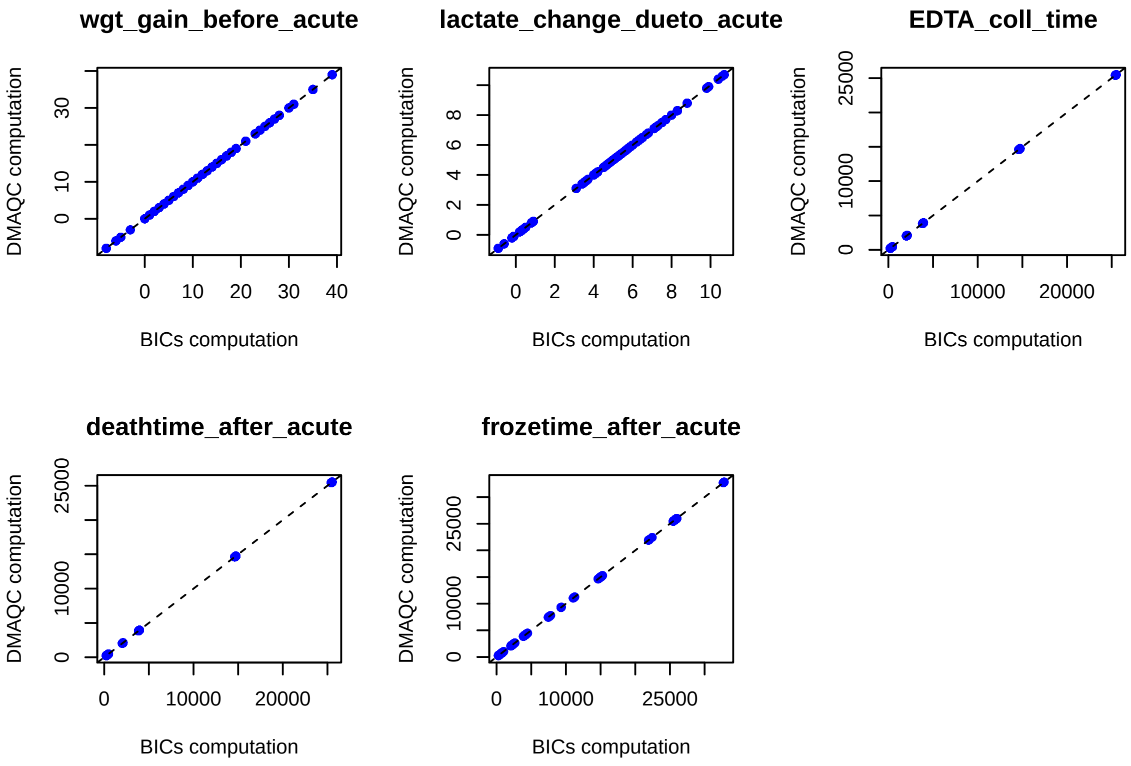
Below, we show that our merged table and computations in R result in the same numbers.

```
# Read the DMAQC calculated fields (do not use the prev ones from the merge
# for an extra QC)
calc_data_file = all_csvs[grepl("Calculated.Variables",all_csvs)]
calc_data = read.csv(calc_data_file)
rownames(calc_data) = calc_data$labelid
```

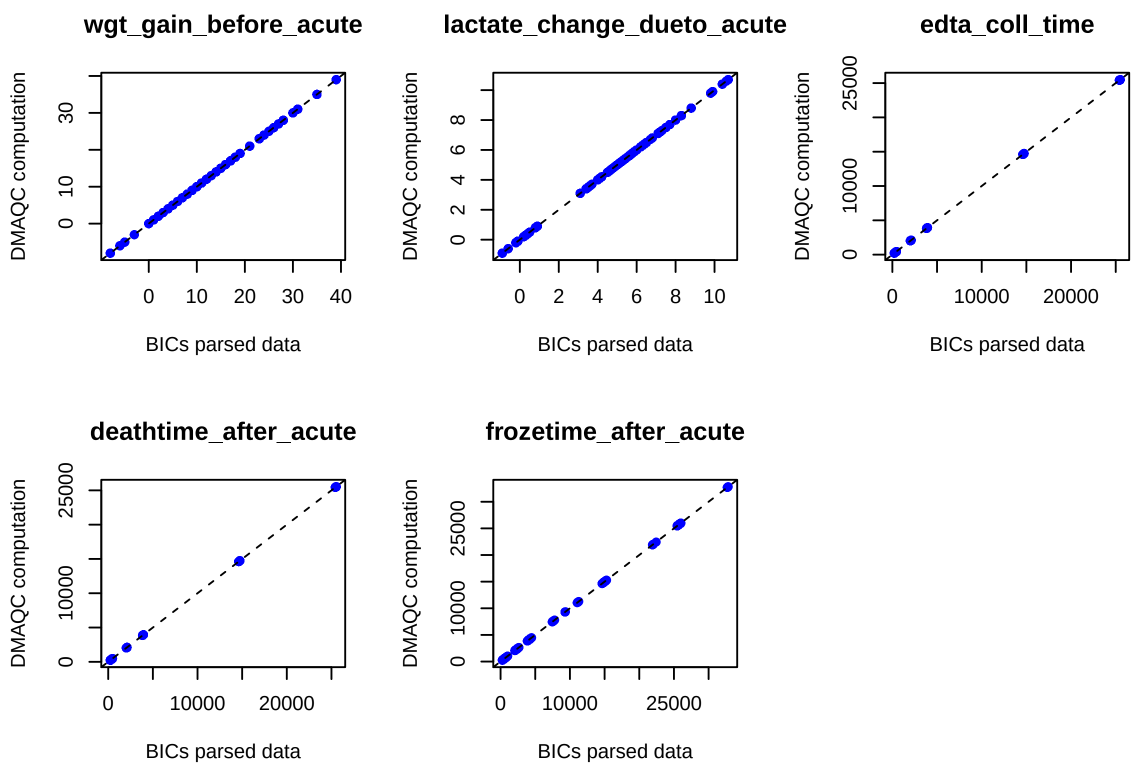
```r
# Extract the relevant columns from our merged dataset
cols_for_analysis = c("labelid",
                      "acute.test.weight","animal.registration.weight",
                      "acute.test.endblood","acute.test.beginblood",
                      "specimen.collection.t_edtafill","acute.test.t_complete",
                      "specimen.collection.t_death","acute.test.t_complete",
                      "specimen.processing.t_freeze","acute.test.t_complete")
# table(is.element(cols_for_analysis,set=colnames(merged_dmaqc_data))) # sanity

# Go over each score and compare the two versions
par(mfrow=c(2,3))
for(j in seq(2,length(cols_for_analysis),by=2)){
  bic_version = unique(merged_dmaqc_data[,cols_for_analysis[c(1,j,j+1)]])
  rownames(bic_version) = bic_version[,1]
  dmaqc_version = calc_data[rownames(bic_version),c(3,3+j/2)]
  if(mode(bic_version[,2])=="character"){
    bic_version_score = as.difftime(bic_version[,2])-as.difftime(bic_version[,3])
    bic_version_score = as.numeric(bic_version_score)*60*60
  }
  else{
    bic_version_score = bic_version[,2]-bic_version[,3]
  }
  plot(bic_version_score,dmaqc_version[,2],pch=20,cex=1.2,col="blue",
       xlab="BICs computation",ylab="DMAQC computation",
       main = colnames(dmaqc_version)[2])
  abline(0,1,lty=2)
}

print("Now go over the columns, but this time take our version from the merged data")
```

```
## [1] "Now go over the columns, but this time take our version from the merged data"
```

```r
colnames(calc_data) = tolower(colnames(calc_data))
par(mfrow=c(2,3))
```



```
for(feature_name in colnames(calc_data)[-c(1:3)]){
  bic_feature_name = paste("calculated.variables.",feature_name,sep="")
  bic_version = unique(merged_dmaqc_data[,c("labelid",bic_feature_name)])
  rownames(bic_version) = as.character(bic_version[,1])
  dmaqc_version = calc_data[rownames(bic_version),c("labelid",feature_name)]
  plot(bic_version[,2],dmaqc_version[,2],pch=20,cex=1.2,col="blue",
       xlab="BICs parsed data",ylab="DMAQC computation",
       main = colnames(dmaqc_version)[2])
  abline(0,1,lty=2)
}
```



6 Correlations with time points

Based on the analyses above we know that the distances are mostly correlated with the shock length and weight/sex. We now plot the achieved distances as a function of the shock data but colored by the time point of each animal in the exercise group.

```
library(ggplot2)
parse_timepoint<-function(x){
  arrs = strsplit(x,split=" ")
  tps = sapply(arrs,function(x)x[3])
  tps = as.numeric(tps)
  tps[is.na(tps)]=-1
  return(tps)
}

# colnames(merged_dmaqc_data)[grepl("sex",colnames(merged_dmaqc_data))]
merged_dmaqc_data$animal.key.timepoint = parse_timepoint(
  merged_dmaqc_data[,"animal.key.anirandgroup"])
```

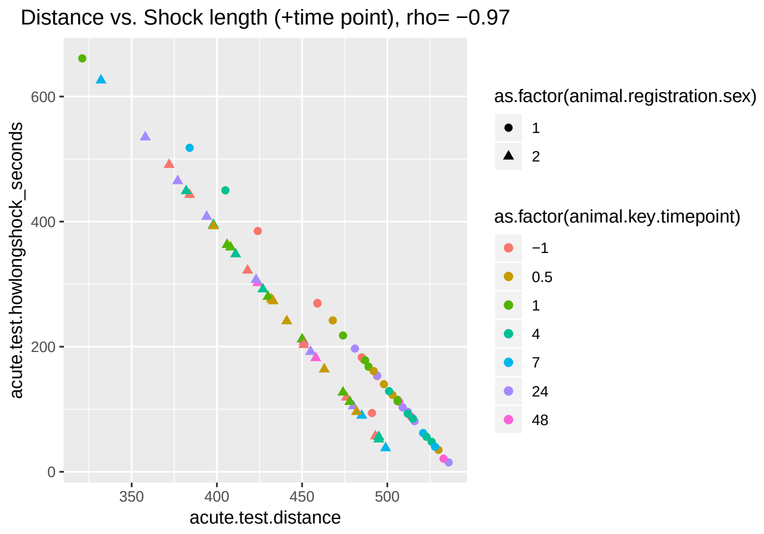
```
## Warning in parse_timepoint(merged_dmaqc_data[, "animal.key.anirandgroup"]):
## NAs introduced by coercion
```

```
merged_dmaqc_data$animal.key.is_control = grepl("control",
      merged_dmaqc_data[,"animal.key.anirandgroup"],ignore.case = T)

# Reduce the data by label ids to avoid duplications
merged_dmaqc_data$acute.test.howlongshock_seconds = sapply(
  merged_dmaqc_data$acute.test.howlongshock,
  parse_shocktime)
```

```
inds = !is.na(merged_dmaqc_data$acute.test.howlongshock_seconds)
df = merged_dmaqc_data[inds,c("bid","acute.test.distance",
                              "acute.test.howlongshock_seconds",
                              "animal.key.timepoint",
                              "animal.registration.sex")]
df = unique(df)
print(paste("Number of bids in the reduced data.",nrow(df)))
```

```
## [1] "Number of bids in the reduced data. 72"
```

```
# Marginal correlation
rho = cor(df$acute.test.howlongshock_seconds,
          df$acute.test.distance)
rho = format(rho,digits = 3)

# A simple 2D plot
ggplot(df,
       aes(x=`acute.test.distance`, y=acute.test.howlongshock_seconds,
           shape=as.factor(animal.registration.sex), color=as.factor(animal.key.timepoint))) +
  geom_point(size=2) + ggtitle(paste("Distance vs. Shock length (+time point), rho=",rho)) +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# Look at the linear regression, do we see a correlation between time
# and distance?
dist_lm2 = lm(acute.test.distance~acute.test.howlongshock_seconds+
                as.factor(animal.key.timepoint)+animal.registration.sex,data=df)
print("No significant linear association between the time points and distance:")
```

```
## [1] "No significant linear association between the time points and distance:"
```

```
summary(dist_lm2)
```

```
##
## Call:
## lm(formula = acute.test.distance ~ acute.test.howlongshock_seconds +
##     as.factor(animal.key.timepoint) + animal.registration.sex,
##     data = df)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max 
## -19.7934  -0.4604   0.6045   1.3591   5.7995 
## 
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       562.059277   1.806668 311.103   <2e-16
## acute.test.howlongshock_seconds    -0.296516   0.003231 -91.770   <2e-16
## as.factor(animal.key.timepoint)0.5   1.901423   1.599209   1.189    0.239
## as.factor(animal.key.timepoint)1    -0.134233   1.589311  -0.084    0.933
## as.factor(animal.key.timepoint)4     2.088966   1.595693   1.309    0.195
## as.factor(animal.key.timepoint)7     1.711643   1.947389   0.879    0.383
## as.factor(animal.key.timepoint)24    2.571109   1.591651   1.615    0.111
## as.factor(animal.key.timepoint)48    1.929356   1.971338   0.979    0.331
## animal.registration.sex           -25.134776   0.971100 -25.883   <2e-16
## 
## (Intercept)                       ***
## acute.test.howlongshock_seconds   ***
## as.factor(animal.key.timepoint)0.5    
## as.factor(animal.key.timepoint)1      
## as.factor(animal.key.timepoint)4      
## as.factor(animal.key.timepoint)7      
## as.factor(animal.key.timepoint)24     
## as.factor(animal.key.timepoint)48     
## animal.registration.sex           ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 3.893 on 63 degrees of freedom
## Multiple R-squared:  0.9949, Adjusted R-squared:  0.9943 
## F-statistic:  1538 on 8 and 63 DF,  p-value: < 2.2e-16
```

7 Save the merged datasets in the cloud

```r
# To see the bucket list
# gsutil ls -p motrpac-portal-dev
write.table(merged_dmaqc_data,file="merged_dmaqc_data.txt",
            quote = F,sep="\t",row.names = F)
save(merged_dmaqc_data,file="merged_dmaqc_data.RData")
write.table(merged_column_dictionary,file="merged_column_dictionary.txt",
            quote = F,sep="\t",row.names = F)
system(paste("~/google-cloud-sdk/bin/gsutil", "cp merged_dmaqc_data.txt",
             "gs://bic_data_analysis/pass1a/pheno_dmaqc/"))
system(paste("~/google-cloud-sdk/bin/gsutil", "cp merged_dmaqc_data.RData",
             "gs://bic_data_analysis/pass1a/pheno_dmaqc/"))
system(paste("~/google-cloud-sdk/bin/gsutil", "cp merged_column_dictionary.txt",
             "gs://bic_data_analysis/pass1a/pheno_dmaqc/"))
system("rm merged_dmaqc_data.txt")
system("rm merged_dmaqc_data.RData")
system("rm merged_column_dictionary.txt")
```