Popgen_Data_Analysis

Anthony Kwong (a1708050)

10/02/2020

This is a tutorial for analysing some simulated data using the popgen.tools R package and discoal.

We will be analysising the toy_data. It is a list of 2000 discoal simulation objects. 1000 are neutral simulations and 1000 are hard sweeps. Half of the hard sweeps have s=0.01 (strong selection) and the other half s=0.001 (weak selection). The sweeps fixed one generation before sampling (present). We used a simple demographic model with a constant effective population size, mutation rate and recombination rate. 200 samples from the present are taken. Below are the simulation parameter for reference.

```
#population params----
mu=1.5e-8
recomb_rate=1e-8
Ne=1e4
nBases=1e6
samplesize=200
s=c(0.001,0.01)
fix=1
discoal_path="~/work/programs/discoal/discoal"
```

```
toy_data<-readRDS("~/work/MPhil/data/toy_data.rds")
```

```
pacman::p_load(popgen.tools,ggplot2,tidyverse)
```

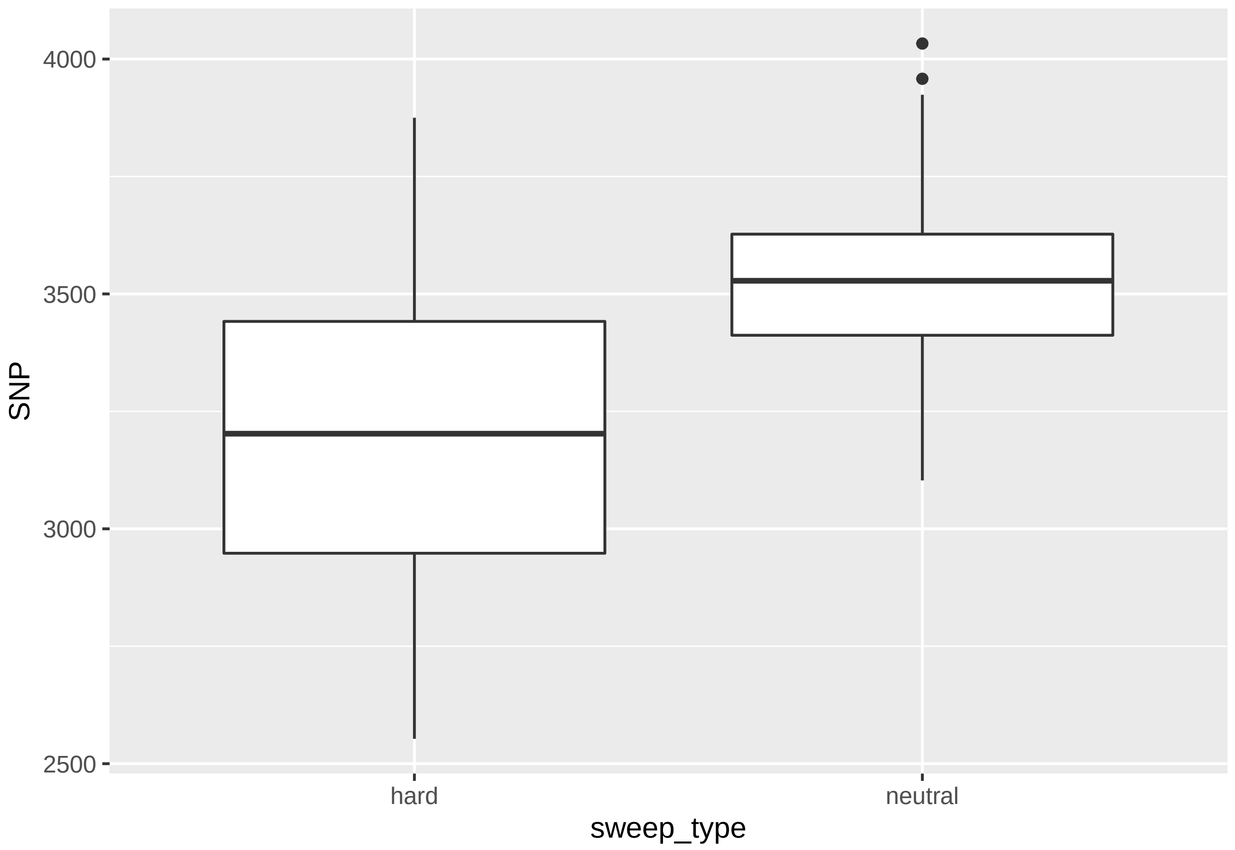
Check SNP distribution

Before we generate our dataframe, let's check the distribution of SNPs across the simulations. One concern is that selective sweeps have lower diversity and hence less SNPs than neutral simulations. However, in real data, there are many other factors for why a particular region may have fewer SNPs, independent of selection. Thus we don't want our method to simply flag regions as a hard sweep based on the number of SNPs. Instead we want to fit our model, using data where the number of SNPs is similar between sweeps and neutral simulations. We also want to see if the data looks sensible before proceeding.

We obtain the snp distribution using snp_count().

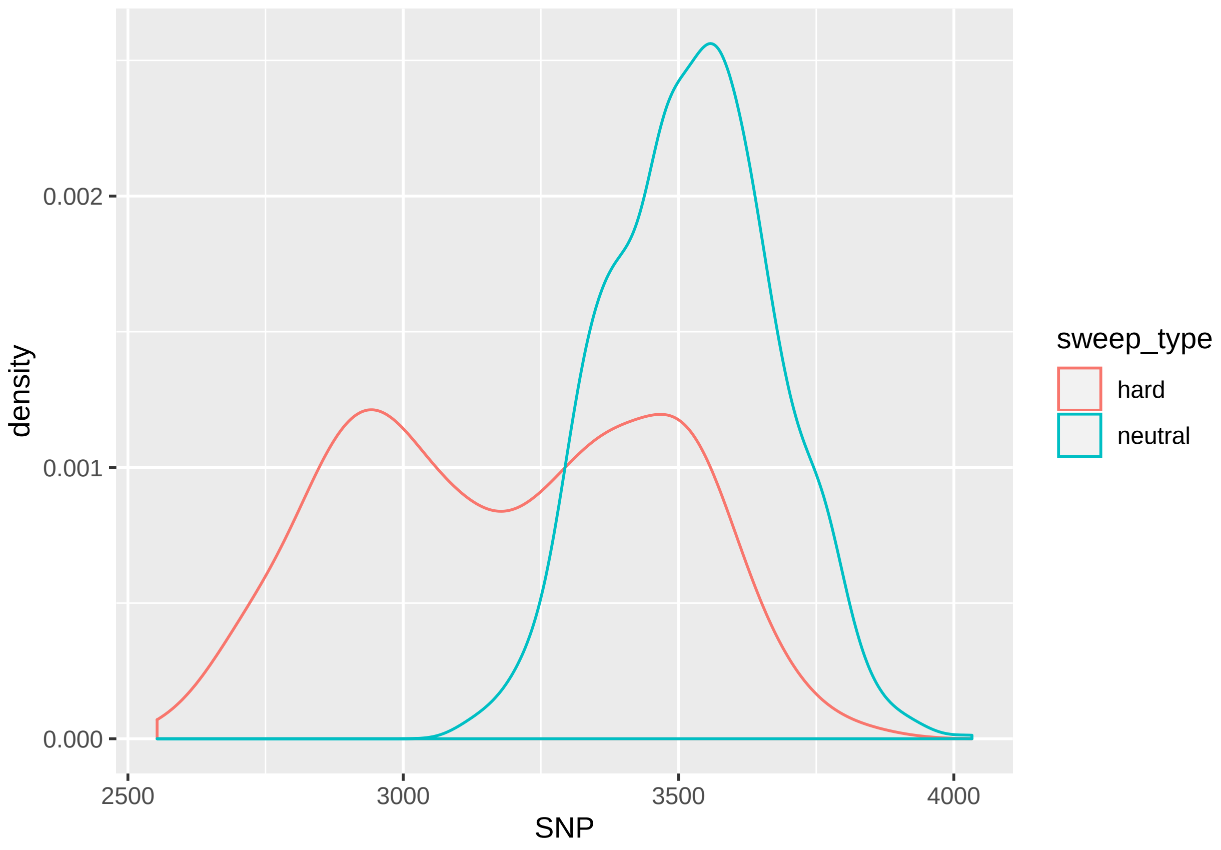
```
#check SNP distribution
snp_dist<-snp_count(toy_data)
```

```
#check snp distribution using boxplots
ggplot(snp_dist,aes(sweep_type,SNP))+geom_boxplot()
```



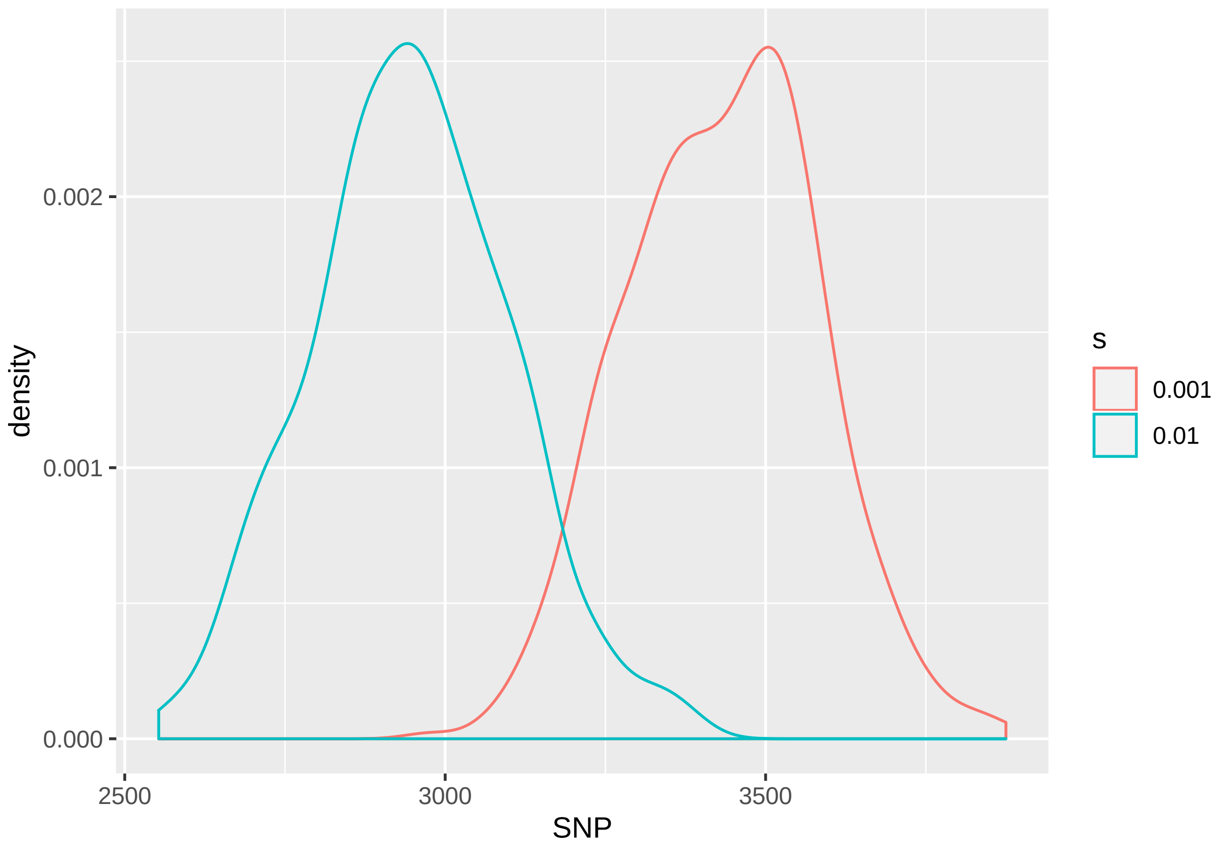
Hard sweeps have lower diversity than neutral simulations, hence fewer SNPs. The distribution is wider for hard sweeps because we used 2 selection coefficients.

```
ggplot(data=snp_dist, aes(x=SNP, color=sweep_type))+ geom_density()
```



The SNP distribution is roughly normal for neutral simulations. Hard sweeps is bimodal. This reflects half the hard sweeps being strong selection and half being weak selection.

```
temp<-subset(snp_dist,sweep_type=="hard")
temp$s<-as.factor(temp$s)
ggplot(data=temp, aes(x=SNP, color=s))+ geom_density()
```



Stronger selection leads to lower diversity and hence fewer SNPs.

Since the hard sweeps with strong selection have the fewest SNPs, we will use its distribution to decide on a SNP cutoff. Let's try a cutoff value of the mean - 1 standard deviation.

```
strong_dist<-temp %>% subset(s==0.01)
avg<-mean(strong_dist$SNP) %>% floor()
std<-sd(strong_dist$SNP) %>% floor()
cutoff<-avg-std
```

```
nrow(subset(snp_dist,SNP>cutoff))/nrow(snp_dist)
```

```
## [1] 0.9605
```

~96% of the simulations have a SNP count higher than the cutoff (2792 SNPs).

Generating the dataframe

```
#df<-generate_df(sim_list = data,win_split =11,snp=cutoff,form="wide")
```

We obtain our dataframe using generate_df(). For all the simulations with more SNPs than the cutoff, we retain the central 2792 SNPs with the selected mutation (if present) in the middle. The dataframe will be in wide form meaning that each row is an individual simulation.

```
df<-read_csv("../data/toy_df.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   sweep = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```r
#data cleaning
df<-as_tibble(df)
df$sweep<-df$sweep %>% as.factor()
df<-subset(df,select=-ID)
```

```r
head(df)
```

```
## # A tibble: 6 x 68
##   sweep s_coef    H_1    H_2    H_3    H_4     H_5  H_6      H_7   H_8    H_9
##   <fct>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>   <dbl> <dbl>    <dbl> <dbl>  <dbl>
## 1 hard   0.001   0.858 -12.7  -25.7   15.8    15.8   -6.21  -3.40   -1.50   9.37
## 2 hard   0.001 -19.3    -1.91   7.01 -15.6     5.78  10.6    0.0360 -6.21  13.9
## 3 hard   0.001  14.0     4.41  12.4    4.21 -12.4   -2.13 -13.9    -1.09  11.7
## 4 hard   0.001  -5.53    5.92   6.97   9.33  -0.468  8.79  -1.22    4.10 -14.2
## 5 hard   0.001   3.90    4.64 -14.5    9.32   0.183 -4.78  -1.59   -6.70 -16.1
## 6 hard   0.001  -9.88  -11.7   -2.18   8.34  -1.71  -8.64   5.55    4.47  -1.88
## # ... with 57 more variables: H_10 <dbl>, H_11 <dbl>, D_1 <dbl>, D_2 <dbl>,
## #   D_3 <dbl>, D_4 <dbl>, D_5 <dbl>, D_6 <dbl>, D_7 <dbl>, D_8 <dbl>,
## #   D_9 <dbl>, D_10 <dbl>, D_11 <dbl>, h1_1 <dbl>, h1_2 <dbl>, h1_3 <dbl>,
## #   h1_4 <dbl>, h1_5 <dbl>, h1_6 <dbl>, h1_7 <dbl>, h1_8 <dbl>, h1_9 <dbl>,
## #   h1_10 <dbl>, h1_11 <dbl>, h2_1 <dbl>, h2_2 <dbl>, h2_3 <dbl>, h2_4 <dbl>,
## #   h2_5 <dbl>, h2_6 <dbl>, h2_7 <dbl>, h2_8 <dbl>, h2_9 <dbl>, h2_10 <dbl>,
## #   h2_11 <dbl>, h12_1 <dbl>, h12_2 <dbl>, h12_3 <dbl>, h12_4 <dbl>,
## #   h12_5 <dbl>, h12_6 <dbl>, h12_7 <dbl>, h12_8 <dbl>, h12_9 <dbl>,
## #   h12_10 <dbl>, h12_11 <dbl>, h123_1 <dbl>, h123_2 <dbl>, h123_3 <dbl>,
## #   h123_4 <dbl>, h123_5 <dbl>, h123_6 <dbl>, h123_7 <dbl>, h123_8 <dbl>,
## #   h123_9 <dbl>, h123_10 <dbl>, h123_11 <dbl>
```

Exploratory Data Analysis

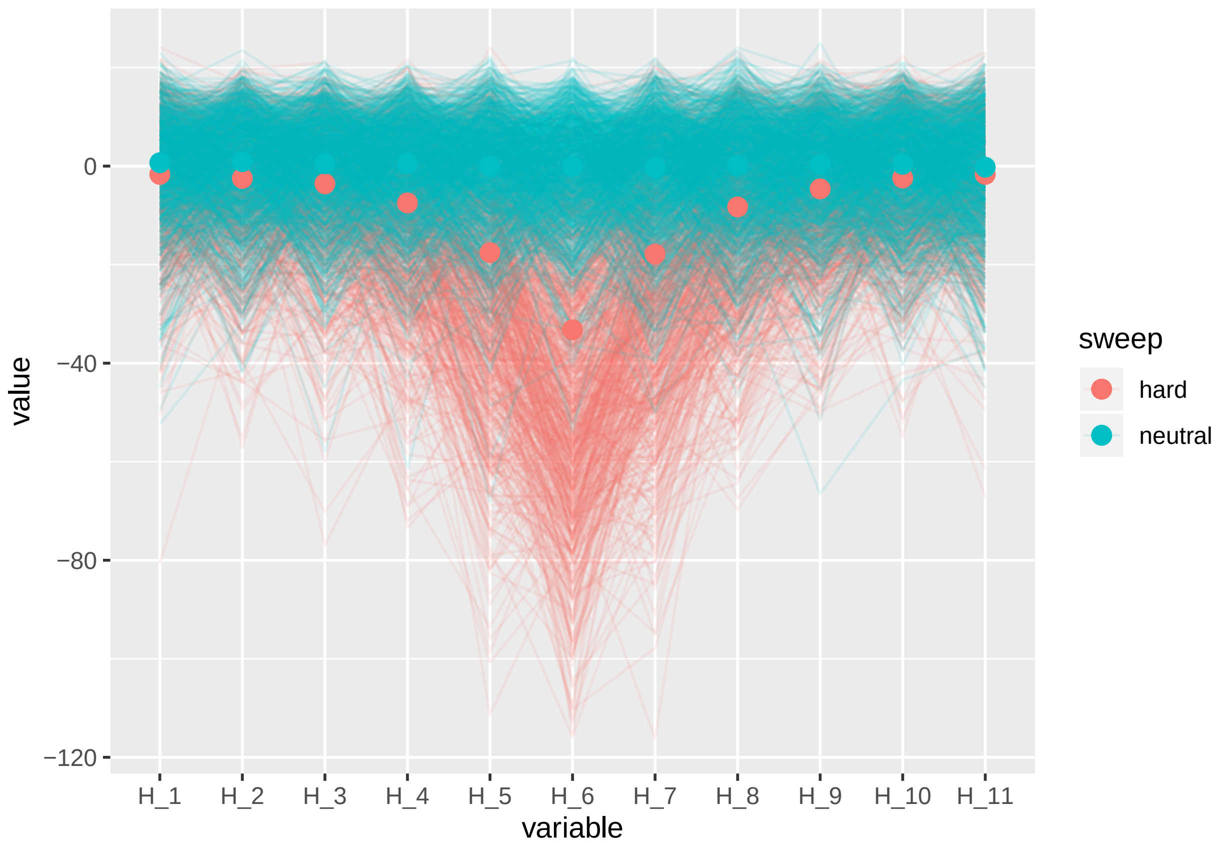
Parallel Coordinates Plot

We will add the mean values onto the parallel coordinates plots.

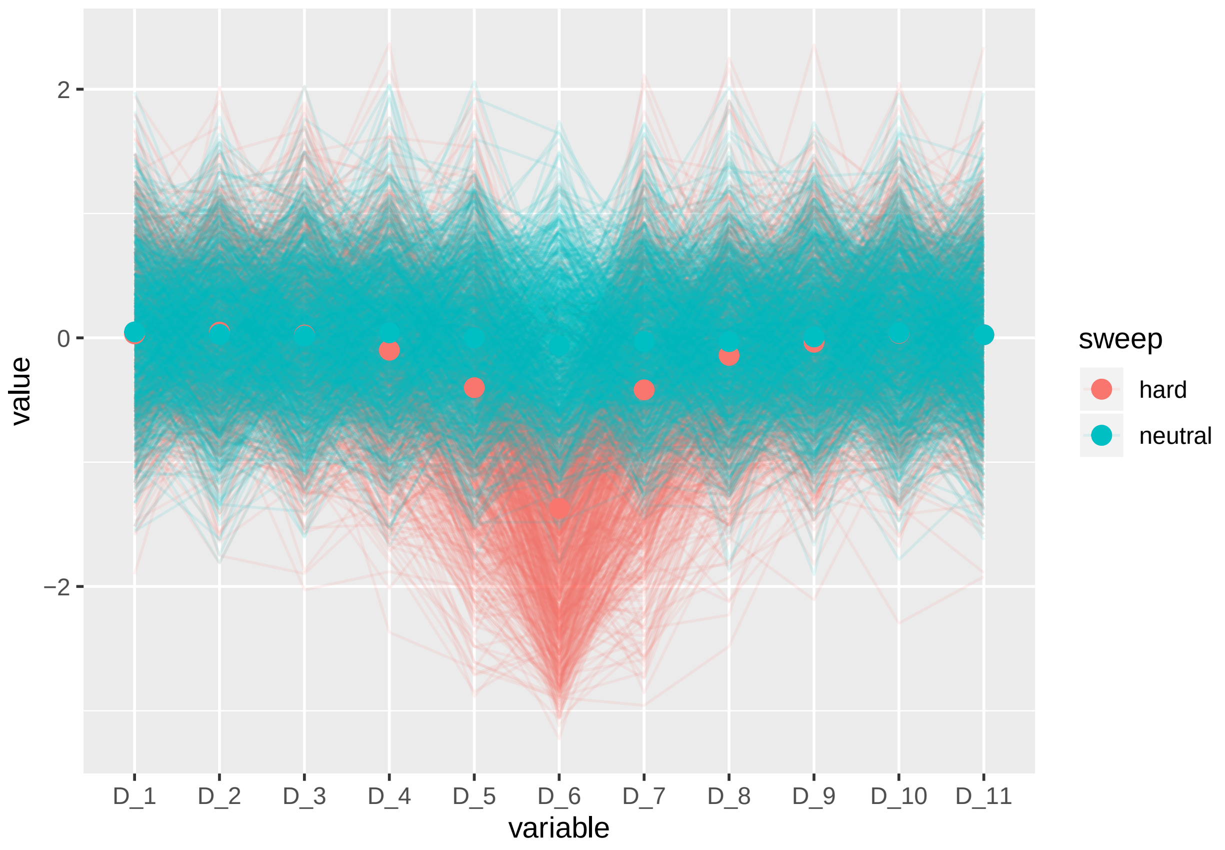
```r
#create df to store means of each variable for both classes
mean_values<- df %>% group_by(sweep) %>%
  summarise_all(mean) %>%
  pivot_longer(-sweep,names_to = "variable", values_to = "value")

#check mean computations
# df %>% dplyr::filter(sweep=="hard") %>% summarise_all(mean)
# df %>% dplyr::filter(sweep=="neutral") %>% summarise_all(mean)
```

```r
pacman::p_load(GGally)
ggparcoord(data=df,columns = (3):(13),groupColumn="sweep",scale="globalminmax",alphaLines = 0.1) +
  geom_point(data=mean_values[2:12,],aes(x=variable, y=value, color=sweep),size=3,inherit.aes = F) +
  geom_point(data=mean_values[69:79,],aes(x=variable, y=value, color=sweep),size=3,inherit.aes = F)
```



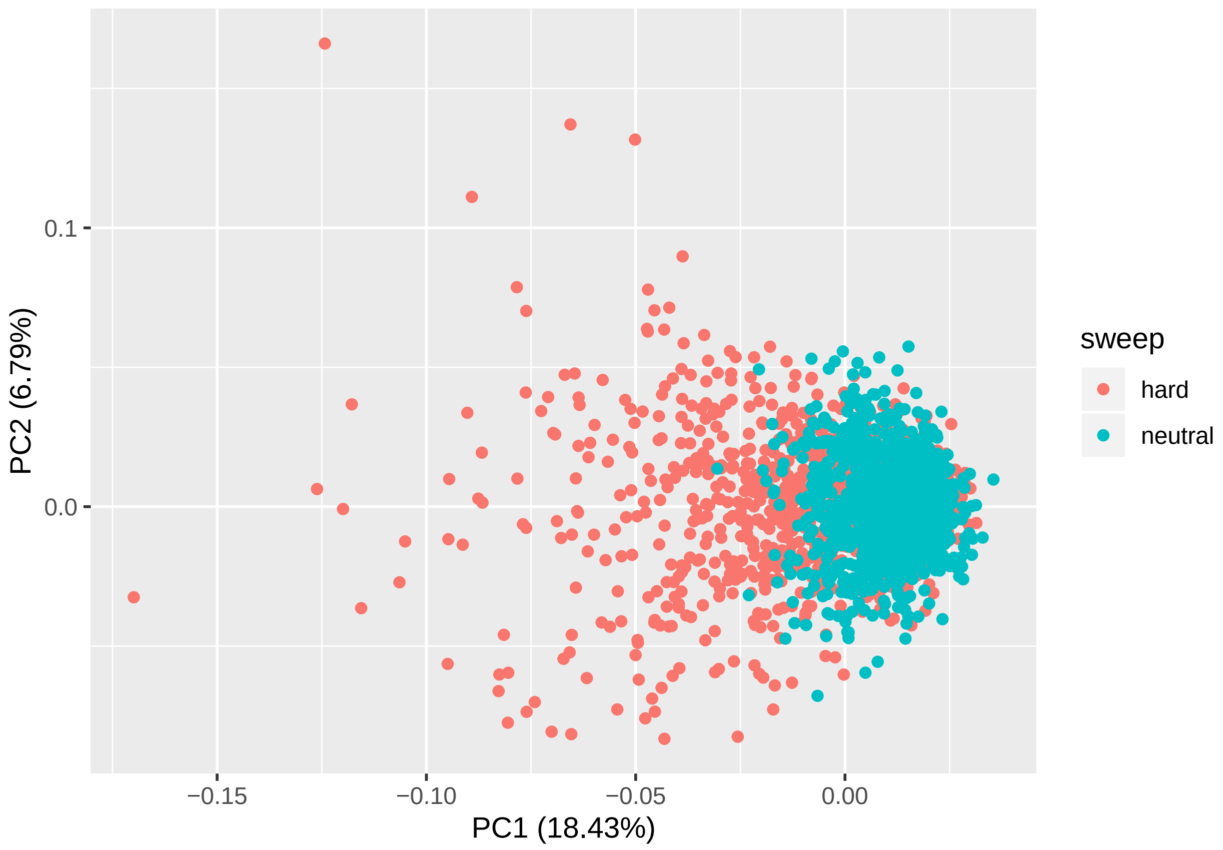
```
ggparcoord(data=df,columns = (14):(24),groupColumn="sweep",scale="globalminmax",alphaLines = 0.1) +
  geom_point(data=mean_values[13:23,],aes(x=variable, y=value, color=sweep),size=3,inherit.aes = F) +
  geom_point(data=mean_values[80:90,],aes(x=variable, y=value, color=sweep),size=3,inherit.aes = F)
```



Both Tajima's D and Fay and Wu's H are measures of diversity. Negative values indicate selection. In both cases, we see the statistic have a trough around window 6. This is because the selected mutation is around the middle of the simulated genomes. The mean of the neutral simulations is ~0 which is the expected result under the neutral model (shown analytically).

PCA

```
pacman::p_load(ggfortify)
#remove response variables for PCA
preds<-select(df,-c(sweep,s_coef))
autoplot(prcomp(preds,scale=T),data=df,colour='sweep')
```



The hard and neutral data do not separate well. There is more variability in hard sweeps along PC1.

Preprocessing

Near Zero Variance Predictors

```
pacman::p_load(caret)
```

Check for near zero variance predictors. These are uninformative and can break some models.

```
nzv<-nearZeroVar(df,saveMetrics = TRUE)
head(nzv)
```

```
##         freqRatio percentUnique zeroVar   nzv
## sweep           1          0.10   FALSE FALSE
## s_coef          2          0.15   FALSE FALSE
## H_1             1         99.65   FALSE FALSE
## H_2             1         99.65   FALSE FALSE
## H_3             1         99.85   FALSE FALSE
## H_4             1         99.80   FALSE FALSE
```

```
#any problematic variables can be removed as such
#filter_df<-df[,-nzv]
```

There are no nzv predictors.

Correlated Predictors

Highly correlated predictors can reduce the performance of some models. We will make the cutoff at 0.6. ie. the highest correlation permitted in the set of predictors is 0.6.

```
df_pred<-select(df,-c(sweep,s_coef))
des_cor<-cor(df_pred)
high_corr<-findCorrelation(des_cor,cutoff = 0.6)
filtered_pred<-df_pred[,-high_corr]
```

Checking our code worked.

```
des_cor2<-cor(filtered_pred)
summary(des_cor2[upper.tri(des_cor2)])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.37992 -0.02493  0.02541  0.03807  0.08575  0.57396
```

Highest correlation in the filtered set of predictors is 0.57. Great the code worked.

Linear Combinations

Remove any linear combinations in the data as they would be redundant.

```
comboInfo<-findLinearCombos(filtered_pred)
#filtered_pred<-filtered_pred[,-comboInfo$remove]
```

No linear combinations found.

Scaling

We will put all the predictors into the same scale so that none would have a disproportionately high effect merely due to the scale it was measured.

```
#substract mean and divide by std for each column. Note that we probably want to do this over a row/win
preProcValues<-preProcess(filtered_pred,method=c("center","scale"))

#make new transformed set of training and test data
dataTransformed<-predict(preProcValues,filtered_pred)
final_tdata<-cbind(df$sweep,dataTransformed)
colnames(final_tdata)[1]<-"sweep"
```

We are now ready to partition the data into training and test sets. We randomly assign 80% of the data for training.

```
set.seed(911939)
indices<-createDataPartition(final_tdata$sweep, times=1, p=0.8,list=FALSE)

train_data<-final_tdata[indices,]
test_data<-final_tdata[-indices,]
```

We can check that both the training and testing datasets have the same proportions for the response variable sweep.

```
prop.table(table(train_data$sweep))
```

```
##
##    hard neutral
##     0.5     0.5
```

```
prop.table(table(test_data$sweep))
```

```
##
##    hard neutral
##     0.5     0.5
```

Model Fitting

We will tune our models using 10 fold cross validation, 3 times for each set of tuning parameters.

```
#do 10 fold CV, 3 times for each model
train.control<-trainControl(method="repeatedcv", number=10, repeats=3,classProbs = TRUE)
```

We will try logistical regression, random forrests, support vector machines and xgboost (gradient descent method).

```
#logistical regression, boosted
grid<-expand.grid(C=seq(5,20,by=5))
lrfit<-train(sweep~., data=train_data, method="LogitBoost",
             nIter=grid,metric="Accuracy",trControl=train.control )

lr.preds<-predict(lrfit, test_data,type="prob")
#confusionMatrix(lr.preds, test_data$sweep)
```

```
#xgboost
tune.grid <- expand.grid(eta = c(0.05, 0.075, 0.1), nrounds = c(50, 75, 100),
                         max_depth = 6,
                         min_child_weight = c(2.0, 2.25, 2.5), colsample_bytree = c(0.3, 0.4, 0.5), gamm
                         subsample = 1)

xg.fit <- train(sweep ~ .,
                data = train_data,
                method = "xgbTree", tuneGrid = tune.grid, trControl = train.control)


xg.preds<-predict(xg.fit,test_data,type="prob")
#confusionMatrix(xg.preds, test_data$sweep)
```

```
#SVM (Linear)
grid<-expand.grid(C=seq(0.5,1.5,by=0.2))
svm.fit<-train(sweep~., data=train_data,method="svmLinear",
               tuneGrid=grid,metric="Accuracy",trControl=train.control)

svm.preds<-predict(svm.fit,test_data,type="prob")
#confusionMatrix(svm.preds,test_data$sweep)
```

```
#Random Forest

grid<-expand.grid(mtry=seq(15,35,by=5))
rf.fit<-train(sweep~.,data=train_data,method="parRF",tuneGrid=grid,metric="Accuracy",trControl=train.co
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
rf.preds<-predict(rf.fit,test_data,type = "prob")

#confusionMatrix(rf.preds,test_data$sweep)
```

Comparing Model Performance

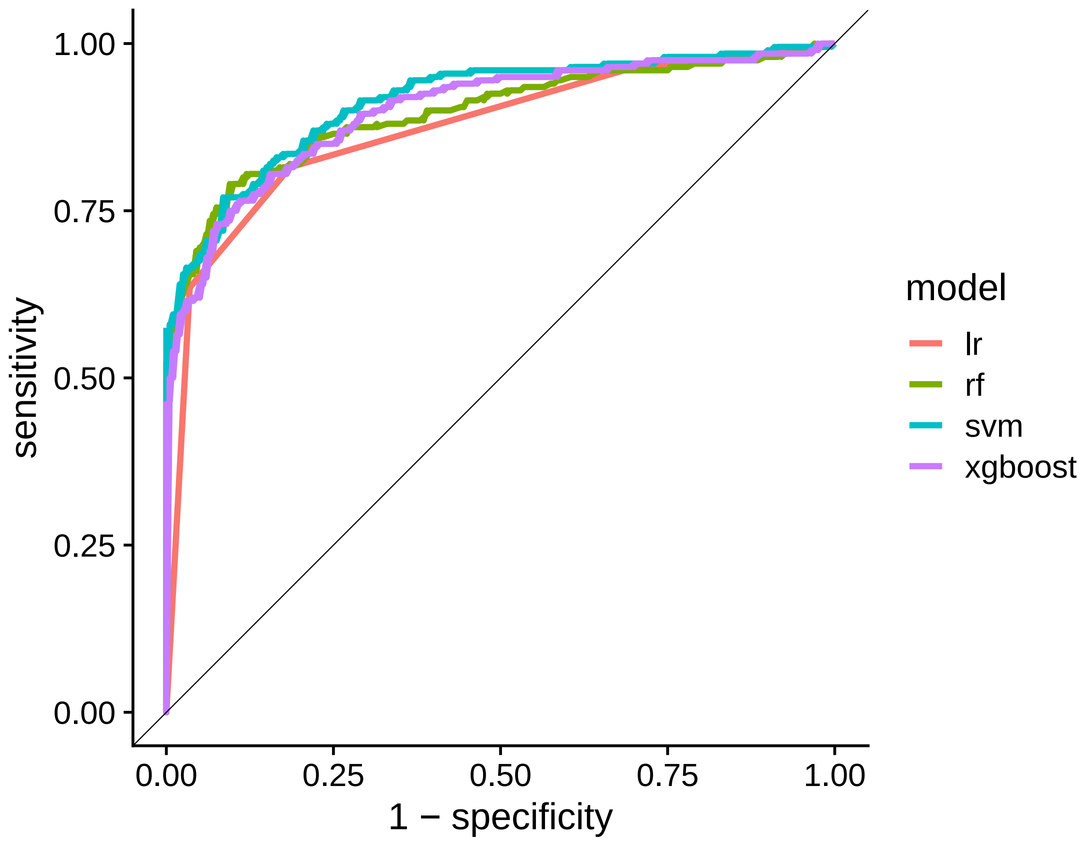
ROC

```r
pacman::p_load(yardstick,cowplot)
#grab the true responses in the test set
truth<-test_data$sweep
model_names<-c("lr","xgboost","svm","rf")
model_preds<-list(lr.preds,xg.preds,svm.preds,rf.preds)

model_eval<-function(model_name,model_pred){
  data<-as.data.frame(model_pred)
  ans<-data %>% mutate(model=model_name) %>% cbind(truth)
  return(ans)
}

model_list<-list()
for(i in 1:length(model_names)){
  model_list[[i]]<-model_eval(model_names[i],model_preds[i])
}
model_out<-plyr::ldply(model_list, data.frame)
```

```r
model_out %>%
  #get individual roc curves for each model
  group_by(model) %>%
  roc_curve(truth=truth, estimate=hard) %>%
  ggplot(
    aes(x=1-specificity,y=sensitivity,color=model)
  ) +
  geom_line(size=1.1) +
  geom_abline(slope=1, intercept = 0, size=0.2) +
  #fix aspect ratio to 1:1 for visualization
  coord_fixed() +
  theme_cowplot()
```



AUC

```
model_out %>%
  group_by(model) %>%
  roc_auc(truth=truth,hard,options = list(smooth = TRUE)) %>%
  knitr::kable()
```

model	.metric	.estimator	.estimate
lr	roc_auc	binary	0.8885666
rf	roc_auc	binary	0.8945084
svm	roc_auc	binary	0.9114272
xgboost	roc_auc	binary	0.8967059