野外 RNA-Seq を用いた品種間差を生み出す座位の検出

龍谷大学　食と農の総合研究所　博士研究員　鹿島　誠

Mail: kashima.biology@gmail.com

2019 年 5 月 28 日

本演習の背景

本解析の目的

植物は、周辺環境に応答して生育している。その環境応答は、ゲノムによって規定されていると考えられ、ゲノム多型が環境応答の品種間差を生み出していると考えられる。遺伝子発現量を量的形質とみなし、関与する量的形質座位である QTL(expression QTL: eQTL)を同定する試みは数多く存在する。eQTL 解析には、同じ環境（同圃場・同時点）での対象とする全系統の遺伝子発現データが必要となる。そのため、労力やコストの問題で一度に検証できる環境の数には限りがあった。

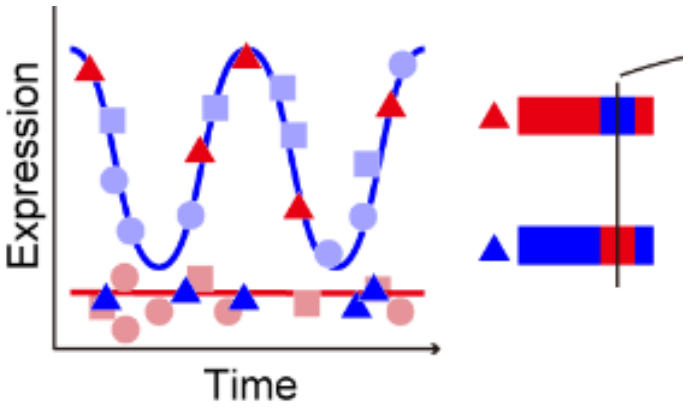
近年、野外の気象条件と継時トランスクリプトームから対象とする系統(genotype)用の数理モデルを作成することで、任意の気象条件での対象系統の遺伝子発現を予測することが可能となっている(Iwayama et al. 2017; Nagano et al. 2012)。

本実習では、この手法を利用して、遺伝子発現動態の品種間差を生み出す座位(expression dynamics QTL: edQTL)を同定し、任意の genotype の植物を予測できる手法が開発された(Kashima et al. 2018)。本演習では、この手法を解説し、気象データと野外で栽培したイネの継時 RNA-Seq のデータから、品種間差を生み出す座位を同定する手法を紹介する。

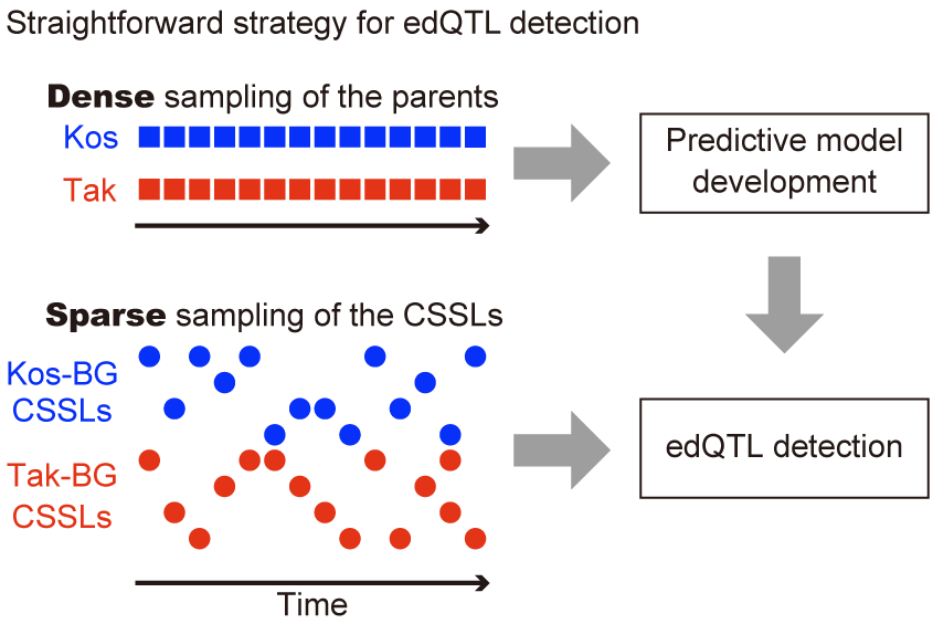
edQTL 検出のコンセプト

下記の手順で edQTL の検出を行う。

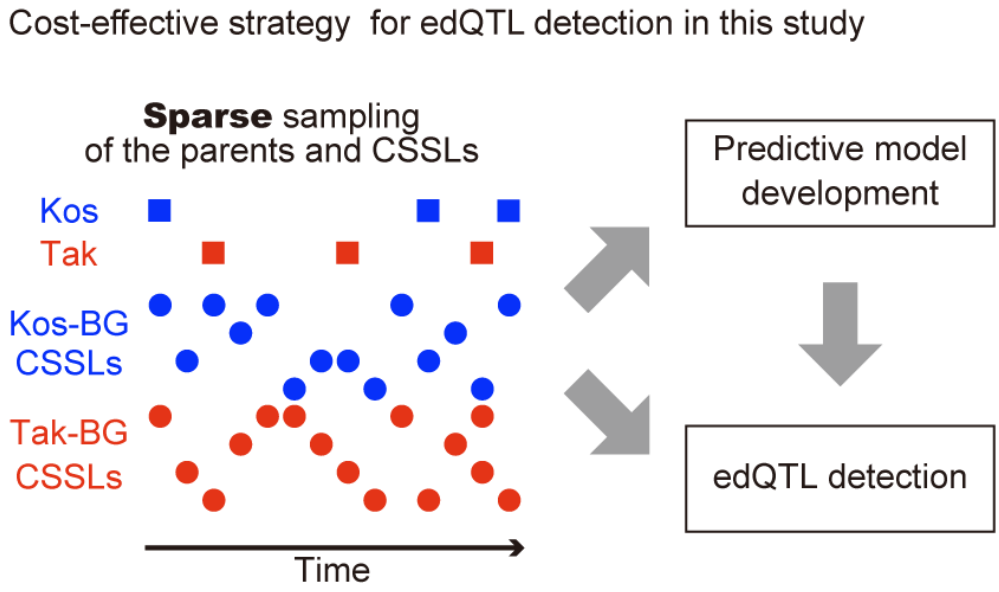
1. 気象条件から遺伝子発現を予測するコシヒカリ用とタカナリ用の予測モデルを作成する。
2. 予測モデルをもとに、各 CSSL のトランスクリプトームの実測値と予測値を比較し、予測値の外れ具合（残差）の大きさと genotype を比較して、edQTL を検出する。



素直な方法としては、コシヒカリ・タカナリを密にサンプリングし、両系統の予測モデルを作成する。その後に、疎にサンプリングした CSSL を用いて edQTL を検出する方法が考えられる。しかし、この方法では、親系統の予測モデル作成にかかる手間とコストが高い。

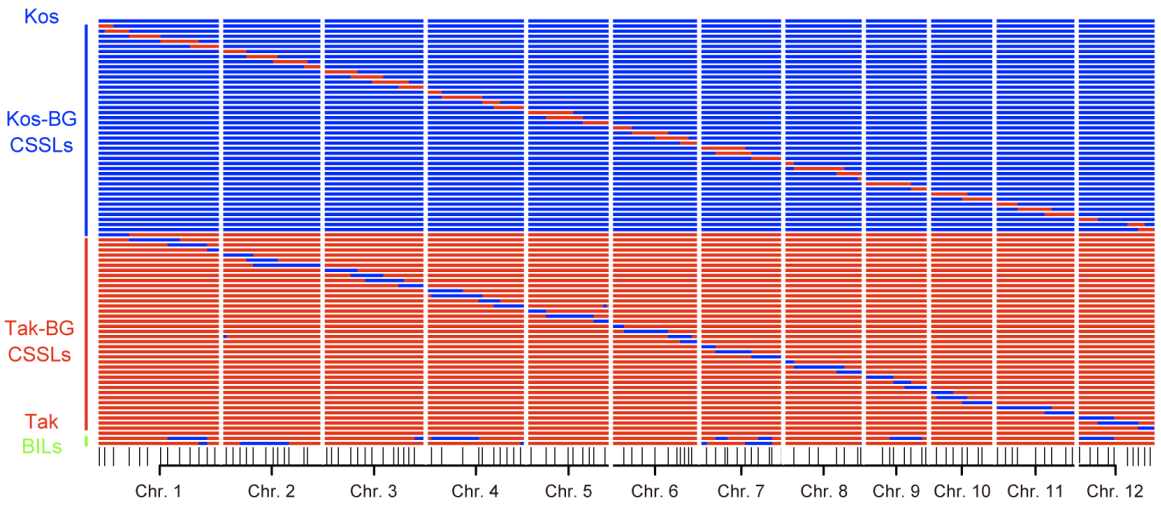


CSSL のゲノムの 95%以上は親系統と同じであり、その遺伝子発現動態のほとんどは親系統と同じであると考えられることを利用して、疎にサンプリングした親系統と同じ背景 genotype を持つ CSSL の両方を利用して、親系統の予測モデルを作成し、その後 edQTL の検出を行う。

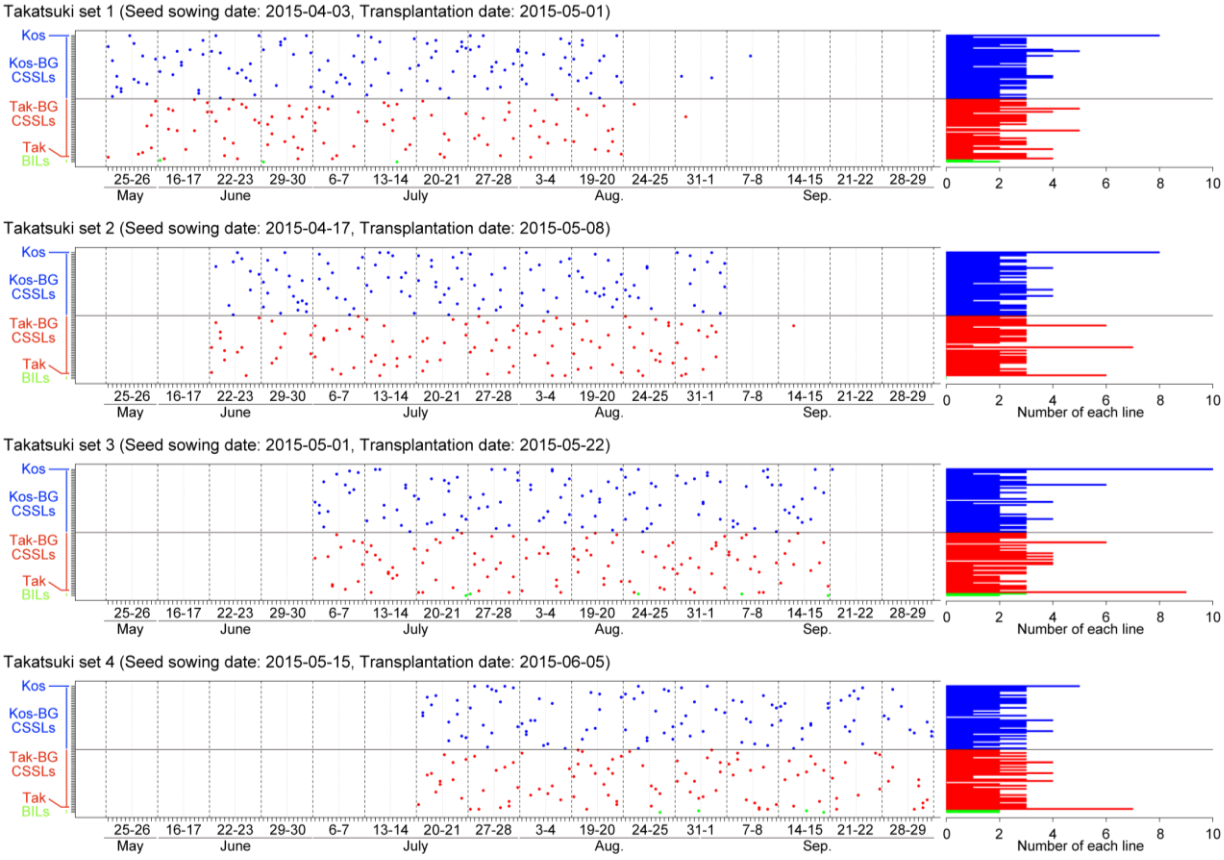


本解析で使用するデータ

今回、対象とする系統は、イネの日本を代表する japonica 品種であるコシヒカリと多収性で近年注目を集めている indica 品種であるタカナリである。加えて、両系統を両親とする染色体断片置換系統(CSSL:Chromosome Segment Substitution Lines)を使用する。以下に、CSSL と戻し交雑自殖系統（BIL:Backcross Inbred Lines）(本演習の最後に使用)の染色体地図を示す。青がコシヒカリ型の染色体を、赤がタカナリ型の染色体を示す。各染色体断片は、141 個の DNA マーカーにより識別されている。



これらを、大阪高槻市の圃場で、四播種期に分けて栽培し、栽培期を通じて計 16 回の最上位展開葉を対象とした二時間毎 24 時間サンプリングを行い、RNA-Seq を行った。下図がサンプリングのまとめとなる。親系統を含めて、偏りなく疎にサンプリングされていることが分かる。

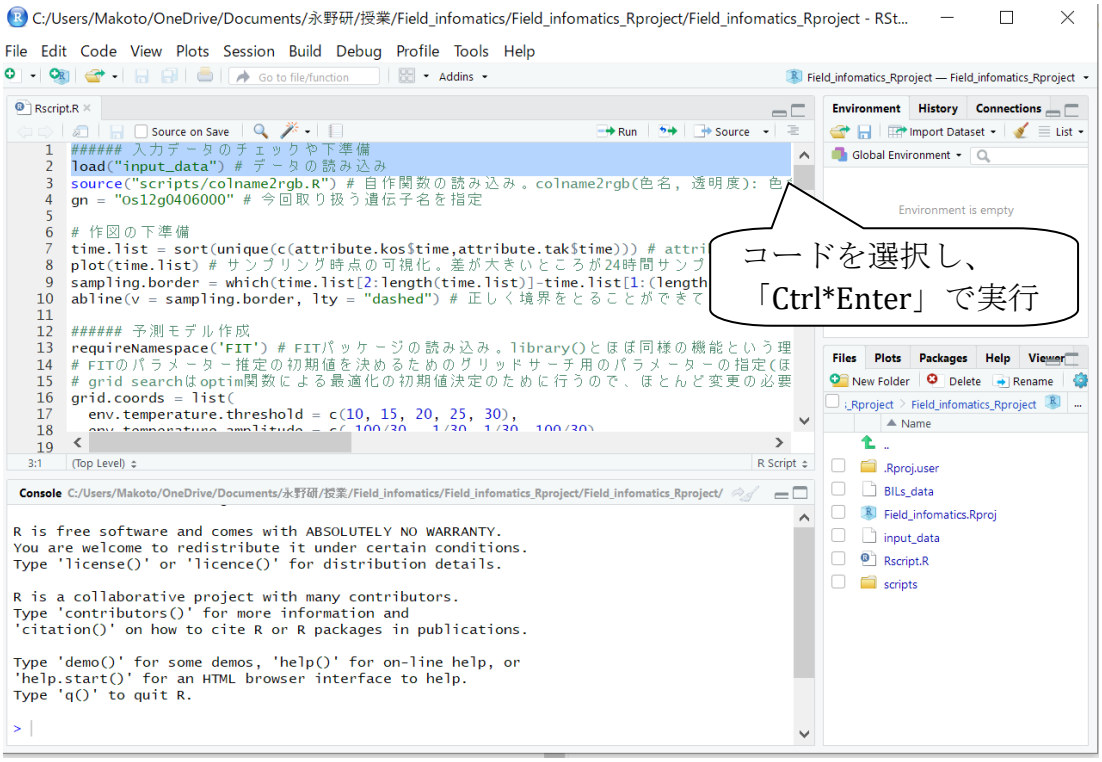


解析コード解説

Rstudio の起動と解析コードの展開

「Field_infomatics_Rproject.zip」を解凍後、解答されたフォルダ内の「Field_informatic.Rproj」を選択して開く。Rstudio が起動するので、続いて「Rscript.R」を開く。

Rstudio の使い方



入力データのチェックや下準備

```
load("input_data") # データの読み込み
source("scripts/colname2rgb.R") # 自作関数の読み込み。colname2rgb(色名, 透明度): 色名と透明度を簡便に指定できる
gn = "Os12g0406000" # 今回取り扱う遺伝子名を指定
```

今回は、時間の都合上、Os12g0406000 に焦点を絞って解析を進める。

読み込んだデータの確認

attribute.kos

	time	sampleID	year	month	day	hour	min	LineName	type	age
20001	208321	20001	2015	5	25	16	0	SL1207	0	37
20003	208441	20003	2015	5	25	18	0	SL1240	0	38
20004	208441	20004	2015	5	25	18	0	SL1217	0	37
20005	208561	20005	2015	5	25	20	0	SL1234	0	37

```
dim(attribute.kos)
```

```
## [1] 447  10
```

```
dim(attribute.tak)
```

```
## [1] 407  10
```

コシヒカリ及びコシヒカリ背景 CSSLs のサンプリング日時、播種後日数、genotype の情報。FIT は time 列を用いてサンプリング時刻の検索をおこなう。2015-01-01 00:00:00 を 0 として、一分ごとに+1。type 列は、通常使用しないので、0 としておく。

gt.mat.full

X	marker	chr.	Position..Mb..IRGSP.1.0	Koshihikari	SL1201	SL1202	SL1203	SL1204	SL1205
1	RM3252	1	0.303	A	B	A	A	A	A
2	RM5423	1	2.170	A	B	B	A	A	A
3	GN1a	1	5.273	A	B	B	A	A	A
4	RM1287-1	1	10.840	A	A	B	B	A	A

各 genotype に対応した 141 DNA マーカーのコシタカ対応表。

gt.mat

Koshihikari	SL1201	SL1202	SL1203	SL1204	SL1205	SL1206	SL1207	SL1208	SL1209
A	B	A	A	A	A	A	A	A	A
A	B	B	A	A	A	A	A	A	A
A	B	B	A	A	A	A	A	A	A
A	A	B	B	A	A	A	A	A	A

```
dim(gt.mat)
```

```
## [1] 141  84
```

各 genotype に対応した 141 DNA マーカーのコシタカ対応表。解析に使用しやすく整形したもの。

log2rpm.kos

	Osm1g00110	Osm1g00120	Osm1g00130	Osm1g00140	Osm1g00150	Osm1g00160
20001	10.039827	9.238876	5.282357	8.352438	-3.321928	-3.321928
20003	10.435200	9.426667	6.878275	8.395283	-3.321928	-3.321928
20004	10.161790	9.242535	6.551691	7.155632	-3.321928	2.299059
20005	9.912023	9.581236	6.390161	6.865035	1.847090	-3.321928

```
dim(log2rpm.kos)
```

```
## [1]   447 23924
```

```
dim(log2rpm.tak)
```

```
## [1]   407 23924
```

各サンプル、各遺伝子の log2 read per million の値。

weather

time	year	month	day	hour	min	temperature	radiation
1	2015	1	1	0	0	5.4	0
2	2015	1	1	0	1	4.5	0
3	2015	1	1	0	2	4.5	0
4	2015	1	1	0	3	4.5	0

```
dim(weather)
```

```
## [1] 525600      8
```

サンプリングを行った一年間の気象情報（気温と全天日射量）。

weights.kos

	Osm1g00110	Osm1g00120	Osm1g00130	Osm1g00140	Osm1g00150	Osm1g00160
20001	0.3414369	0.2923990	0.1637512	0.2124301	0.1272098	0.1545012
20003	0.4933470	0.4149699	0.2219424	0.2943396	0.1153867	0.1202134
20004	0.5721302	0.4771750	0.2507674	0.3357457	0.1200098	0.1146971
20005	0.4942924	0.4130002	0.2213397	0.2936699	0.1153170	0.1203227

```
dim(weights.kos)
```

```
## [1]   447 23924
```

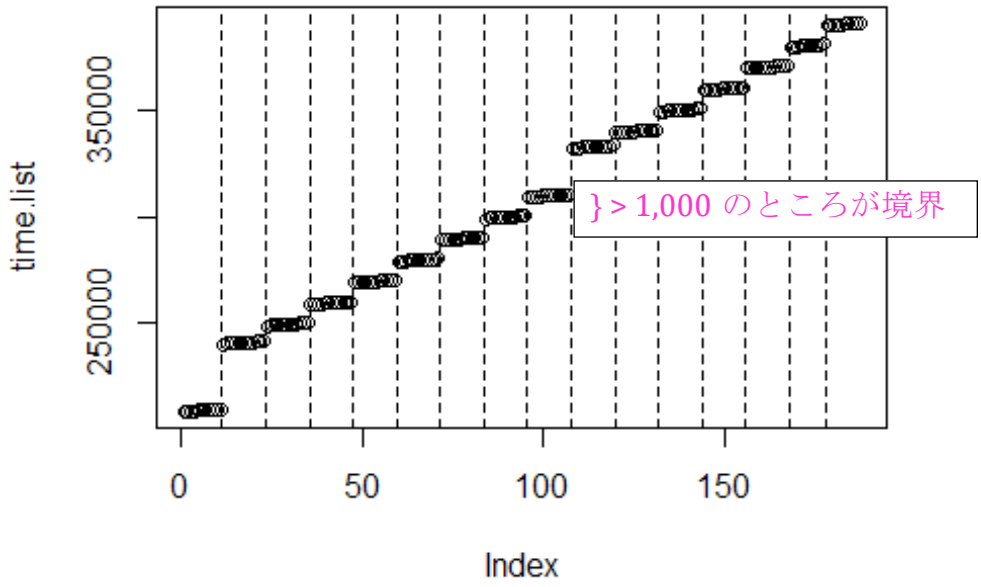
```
dim(weights.tak)
```

```
## [1]   407 23924
```

FIT でのモデル作成時に、重み付け回帰を行うために必要な、raw カウント数から計算した各 log2rpm の信頼度。本実習では、時間の関係で計算方法等の詳細説明は省く。詳しくは、FIT のマニュアルページ(https://cran.r-project.org/web/packages/FIT/vignettes/FIT.html)参照。

作図の下準備

```
time.list = sort(unique(c(attribute.kos$time,attribute.tak$time))) # attribute に存在するすべての time 一覧
plot(time.list) # サンプリング時点の可視化。差が大きいところが 24 時間サンプリングセットの境界。
sampling.border = which(time.list[2:length(time.list)]-time.list[1:(length(time.list)-1)]>1000)+0.5 # which は TRUE の位置を返す
abline(v = sampling.border, lty = "dashed") # 正しく境界をとることができているかをプロットでチェック
```



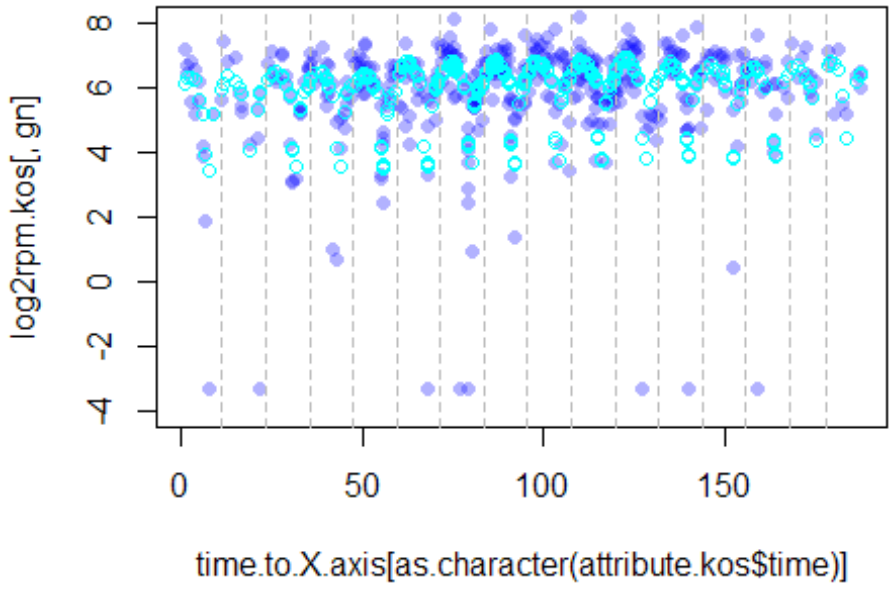
予測モデル作成

```
requireNamespace('FIT') # FIT パッケージの読み込み。Library()とほぼ同様の機能という理解で良いが、こちらは関数の上書きが起こらない。

## Loading required namespace: FIT

# FIT のパラメーター推定の初期値を決めるためのグリッドサーチ用のパラメーターの指定(ほぼデフォルト設定)
# grid search はoptim 関数による最適化の初期値決定のために行うので、ほとんど変更の必要はない。予測結果をみて、問題がある場合に変更する。
grid.coords = list(
  env.temperature.threshold = c(10, 15, 20, 25, 30),
  env.temperature.amplitude = c(-100/30, -1/30, 1/30, 100/30),
  env.radiation.threshold = c(1, 10, 20, 30, 40),
  env.radiation.amplitude = c(-100/80, -1/80, 1/80, 100/80),
  env.temperature.period = c(10, 30, 90, 270, 720, 1440, 1440*3),
  env.radiation.period = c(10, 30, 90, 270, 720, 1440, 1440*3),
  gate.temperature.phase = seq(0, 23*60, 1*60),
  gate.radiation.phase = seq(0, 23*60, 1*60),
  gate.temperature.threshold = cos(pi*seq(8,24,4)/24),
  gate.radiation.threshold = cos(pi*seq(8,24,4)/24),
  gate.temperature.amplitude = c(-5, 5),
  gate.radiation.amplitude = c(-5, 5)
)

# FIT によるモデル作成のためのパラメーター最適化の方法を指定(ほぼデフォルト設定)
recipe = FIT::make.recipe(c("temperature", "radiation"),
```

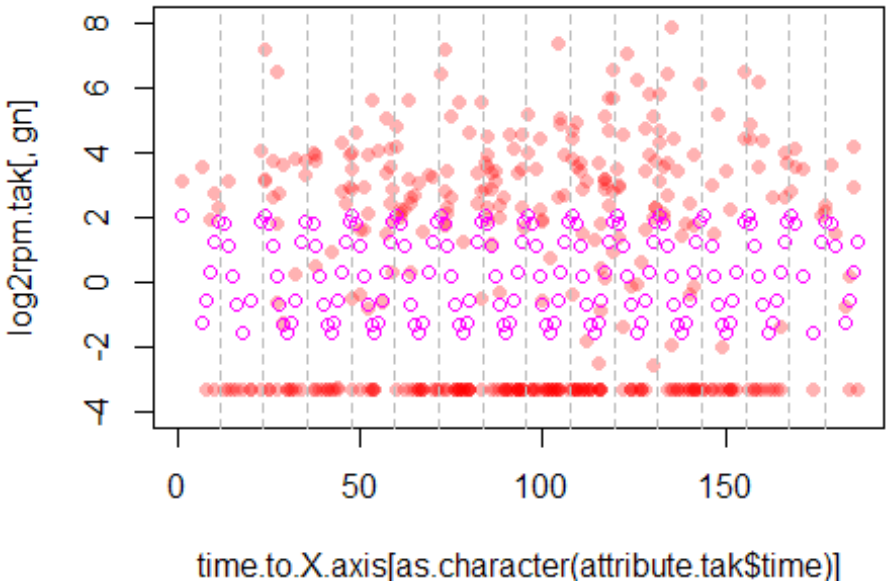
```
                            init = "gridsearch",
                            optim = c("lm"),
                            fit = "fit.lasso",
                            init.data = grid.coords,
                            time.step = 10,
                            gate.open.min = 479)

# コシヒカリ予測モデルの作成
train.attribute.kos  = FIT::convert.attribute(attribute.kos) # attribute
をFIT用に変換

## # Preparing attribute data..done.

train.weather    = FIT::convert.weather(weather, c("temperature","radiati
on")) # weatherをFIT用に変換

## # Preparing weather data..done.

train.expression.kos = FIT::convert.expression(log2rpm.kos, gn) # log2rpm
をFIT用に変換

## # Preparing expression data..done.

train.weights.kos = FIT::convert.weight(weights.kos,gn) # log2rpmに対応し
たweigthをFIT用に変換

## # Preparing weight data..done.

models.kos = FIT::train(train.expression.kos, train.attribute.kos, train.
weather, recipe, train.weights.kos) # FITによるコシヒカリ予測モデル作成

## # * Training..
## # ** Prep+Init:
## # Prep (grids)
## # - D, type, C
## # - E(temperature)
## # - E(radiation)
## # Init (grid search)
## # - init params for temperature
## # - init params for radiation
## # Prep (grids)
## # - D, type, C
## # - E(temperature)
## # - E(radiation)
## # Init (grid search)
## # - init params for temperature
## # - init params for radiation
## # ** Optim (lm):
```

```
## # *** Lm:
## # optimizing Os12g0406000
## # | temperature o | radiation o | => ( temperature , 124.4582 )
## # ** Creating optimized models
## # Done (training)

# タカナリ予測モデルの作成
train.attribute.tak  = FIT::convert.attribute(attribute.tak)

## # Preparing attribute data..done.

train.expression.tak = FIT::convert.expression(log2rpm.tak,gn)

## # Preparing expression data..done.

train.weights.tak = FIT::convert.weight(weights.tak,gn)

## # Preparing weight data..done.

models.tak = FIT::train(train.expression.tak, train.attribute.tak, train.
weather, recipe,train.weights.tak)

## # * Training..
## # ** Prep+Init:
## # Prep (grids)
## # - D, type, C
## # - E(temperature)
## # - E(radiation)
## # Init (grid search)
## # - init params for temperature
## # - init params for radiation
## # Prep (grids)
## # - D, type, C
## # - E(temperature)
## # - E(radiation)
## # Init (grid search)
## # - init params for temperature
## # - init params for radiation
## # ** Optim (lm):
## # *** Lm:
## # optimizing Os12g0406000
## # | temperature o | radiation o | => ( radiation , 483.1129 )
## # ** Creating optimized models
## # Done (training)

## 各サンプリング時点におけるコシヒカリ・タカナリの遺伝子発現予測
kos = FIT::predict(models.kos[[gn]], train.attribute.kos,train.weather) #
 attribute(サンプリング時点と播種後日数)と気象情報から、コシヒカリの遺伝子発現
```

```
を予測

time.to.X.axis = 1:length(time.list) # 未サンプリング期間を飛ばしてplot するため
names(time.to.X.axis) = time.list # attribute のtime からのアクセスを容易にする。

plot(time.to.X.axis[as.character(attribute.kos$time)], log2rpm.kos[,gn], col = colnames2rgb("blue",30), pch = 16, ylim = c(-4,8)) # 実測値をplot。pch=16 は●を指定。
points(time.to.X.axis[as.character(attribute.kos$time)], kos[[1]], col = "cyan") # 予測値をplot
abline(v=sampling.border, lty = "dashed", col = "gray") # 24 時間サンプリングの境界線を描画。
```



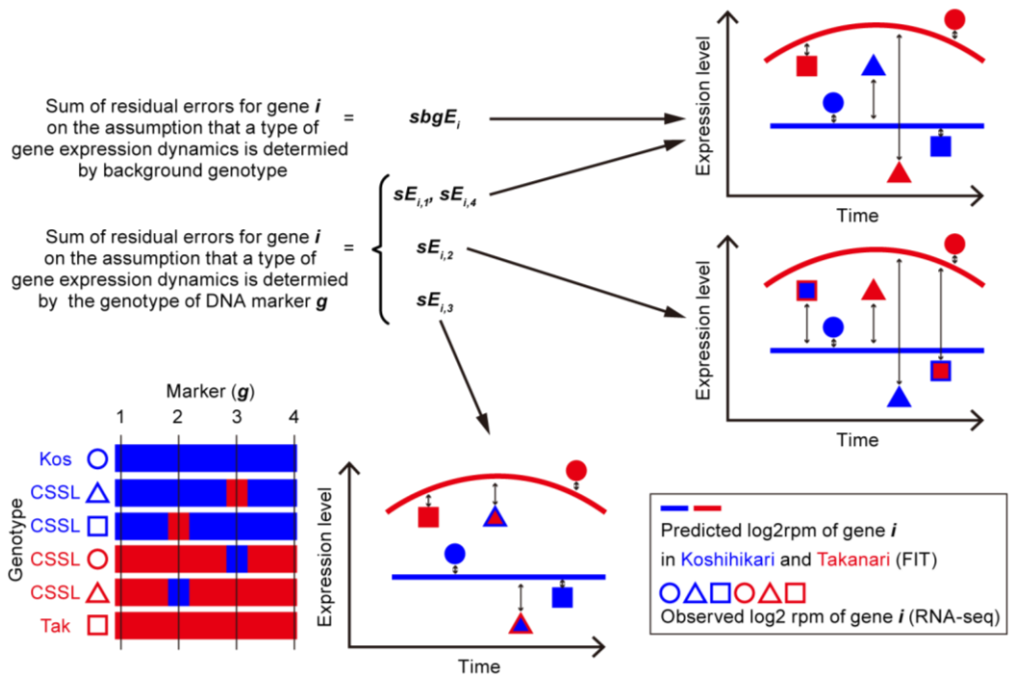

```
# タカナリに関してもplot
tak = FIT::predict(models.tak[[gn]], train.attribute.tak,train.weather)
plot(time.to.X.axis[as.character(attribute.tak$time)], log2rpm.tak[,gn], col = colnames2rgb("red",30), pch = 16, ylim = c(-4,8))
points(time.to.X.axis[as.character(attribute.tak$time)], tak[[1]], col = "magenta")
abline(v=sampling.border, lty = "dashed", col = "gray")
```



コシヒカリの予測は実測値に使いが、タカナリの予測値は実測値よりも低いあるいは高い値になっている。これはタカナリ背景の CSSL ではリードカウントが 0 のサンプル多く、回帰結果がそれに引っ張られたためである。RNA-Seq による定量は、カウントデータであるため、一定値以下の発現量の計測は未検出になってしまうう。この問題は、リード数増やすことで多少は改善される。

edQTL の検出方法解説

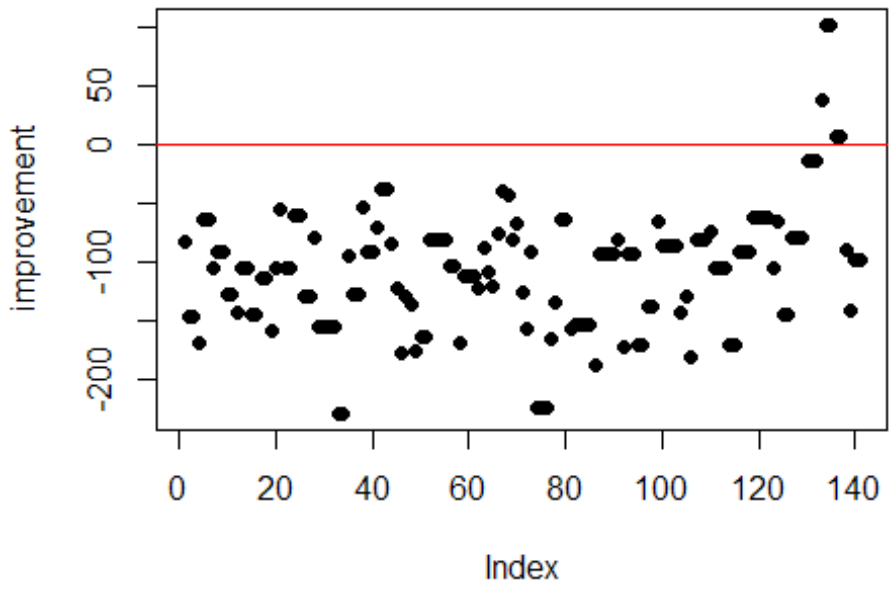
edQTL の検出は、各 DNA マーカーに edQTL があると仮定し、その DNA マーカーの genotype に基づいてコシヒカリモデルかタカナリモデルを選択した場合の残差(予測値の実測値の差)の和が、背景 genotype でモデル選択をした場合の残差和に比べて、どの程度改善するのかを評価することで行う。



edQTL が各 DNA マーカー上に存在すると仮定した際の残差の改善を計算

```
log2rpm.all = rbind(log2rpm.kos,log2rpm.tak)[,gn] # コシヒカリ・タカナリ背景CSSL と親系統の Os12g0406000 の log2rpm を結合
attribute.all = rbind(attribute.kos, attribute.tak) # コシヒカリ・タカナリ背景CSSL と親系統の attribute を結合
prediction.attribute  = FIT::convert.attribute(attribute.all) # attribute.all を FIT の入力用に変換。

## # Preparing attribute data..done.

kos = FIT::predict(models.kos[[gn]], prediction.attribute,train.weather)
# prediction.attribute を入力として、コシヒカリモデルを用いて遺伝子発現を予測
tak = FIT::predict(models.tak[[gn]], prediction.attribute,train.weather)
# prediction.attribute を入力として、タカナリモデルを用いて遺伝子発現を予測

residual.kos = abs(log2rpm.all-kos$temperature) # 実測値とコシヒカリモデルの予測値の残差を計算
residual.tak = abs(log2rpm.all-tak$radiation) # 実測値とタカナリモデルの予測値の残差を計算

sum.of.residual.erros.BG =sum(residual.kos[1:nrow(attribute.kos)],residual.tak[(nrow(attribute.kos)+1):nrow(attribute.all)]) # 背景 genotype に基づいてモデル選択を行い残差和を計算
```

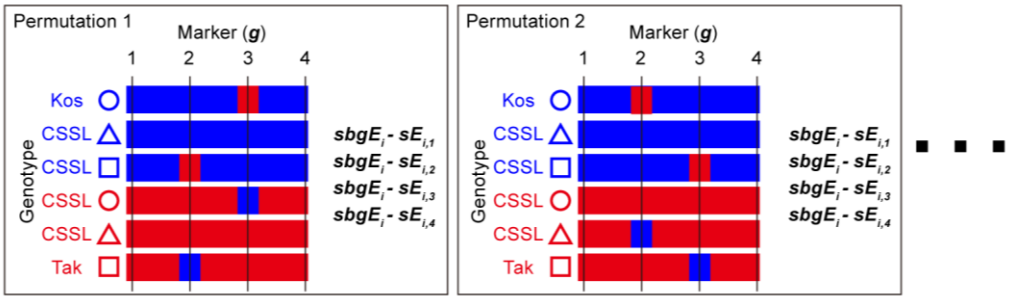
```
sum.of.residual.erros.edQTL = rep(0, nrow(gt.mat)) # 各DNAマーカーにedQTLを仮定した際の残差和の保存用オブジェクト

for(m in 1:nrow(gt.mat)){ # m番目のDNAマーカーにedQTLがあると仮定
 # m = 1
  kos.type = colnames(gt.mat)[gt.mat[m,]=="A"] # edQTLがコシヒカリ型の系統一覧
  tak.type = colnames(gt.mat)[gt.mat[m,]=="B"] # edQTLがタカナリ型の系統一覧
  sample.kos.type = is.element(attribute.all$LineName,kos.type) # edQTLがコシヒカリ型のサンプル一覧
  sample.tak.type = is.element(attribute.all$LineName,tak.type) # edQTLがタカナリ型のサンプル一覧
  sum.of.residual.erros.edQTL[m] = sum(residual.kos[sample.kos.type], residual.tak[sample.tak.type]) # 残差和を計算
}
improvement = sum.of.residual.erros.BG - sum.of.residual.erros.edQTL # 背景genotypeベースと比べた、edQTLベースにおける残差和の改善を計算
plot(improvement, pch = 16) # improvementをplot
abline(h = 0, col = "red") # 改善有無の境界を描画
```



背景 genotype でモデル選択した場合に比べて、残差和が改善しているマーカーを検出できた。

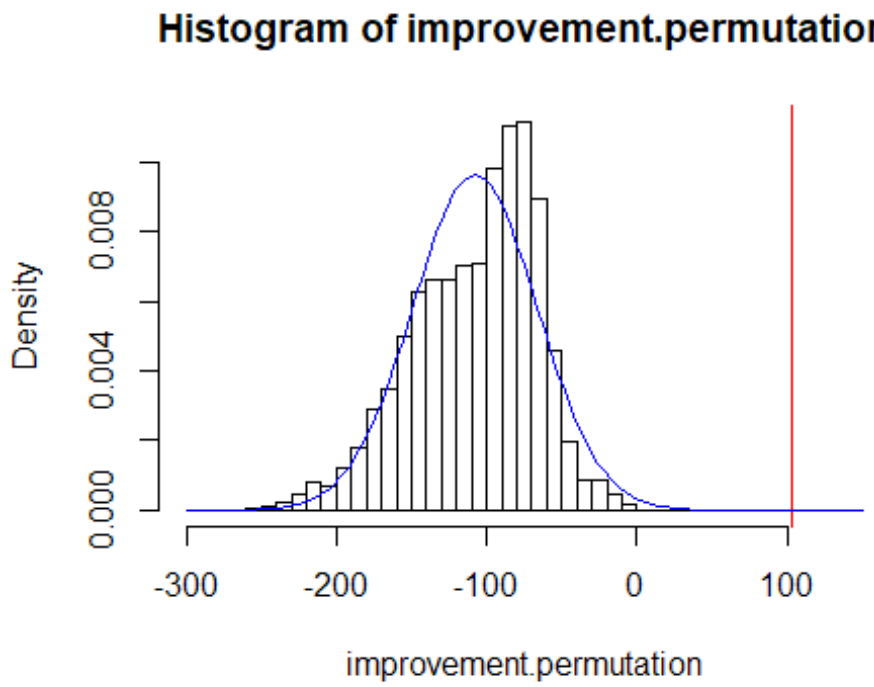
サンプルの genotype に対する permutation を利用した、改善具合の統計的評価

コシヒカリ背景のサンプル、タカナリ背景のサンプル内それぞれで系統名を並び替え(permutation)、残差和の改善を計算することを繰り返すことで、上記で検出した改善がどの程度珍しいのか(統計的優位性)を検討する。



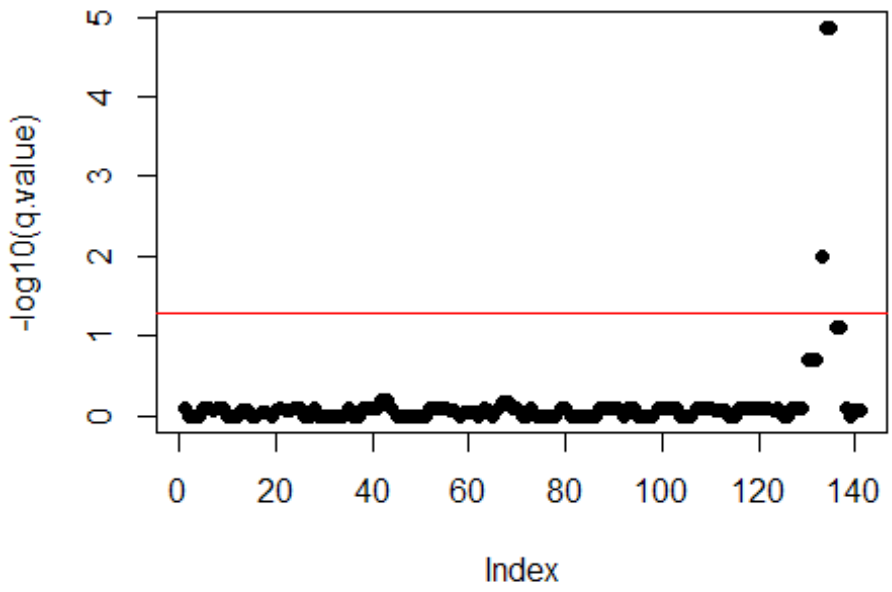

```
set.seed(1234) # permutation の結果の再現性を保障するおまじない
improvement.permutation = matrix(0, ncol = nrow(gt.mat), nrow = 100) # 10
0 回の permuation の結果保存用オブジェクト

# sample 関数の挙動を確認
head(sample(attribute.kos$LineName))

## [1] "SL1205" "SL1216" "SL1205" "SL1206" "SL1208" "SL1218"

head(sample(attribute.kos$LineName))

## [1] "SL1206" "SL1204" "SL1216" "SL1208" "SL1209" "SL1214"

# 100 回の permutation
for(i in 1:100){
  permutated.genotypes = c(sample(attribute.kos$LineName),sample(attribut
e.tak$LineName)) # 各背景 genotype 毎に並べ替え
  for(m in 1:nrow(gt.mat)){ # 各 DNA マーカーに edQTL があると仮定して改善を計
算
    kos.type = colnames(gt.mat)[gt.mat[m,]=="A"] # edQTL がコシヒカリ型の系
統一覧
    tak.type = colnames(gt.mat)[gt.mat[m,]=="B"] # edQTL がタカナリ型の系統
一覧
    sample.kos.type = is.element(permutated.genotypes,kos.type) # Permuta
tion 後のコシヒカリ型のサンプルを選択
    sample.tak.type = is.element(permutated.genotypes,tak.type) # Permuta
tion 後のタカナリ型のサンプルを選択
    improvement.permutation[i,m] = sum.of.residual.erros.BG-sum(residual.
```

```
kos[sample.kos.type], residual.tak[sample.tak.type]) # Permutation 後の系統に基づく、edQTL を仮定することによる残差和の改善を計算
  }
}

which(improvement==max(improvement)) # edQTL を仮定することで、最も残差和が改善する DNA マーカー

## [1] 134 135

m = 134 # DNA マーカーを指定
hist(improvement.permutation, breaks = seq(-300,150,10), freq=F) # 100 回の permutation での残差和の改善の分布
abline(v=improvement[m], col = "red") # 134 の DNA マーカーに edQTL が存在すると仮定した際の残差和の改善を描画
mean = mean(improvement.permutation) # permutation で得られた残差和の改善の平均を計算
sd = sd(improvement.permutation) # permutation で得られた残差和の改善の標準偏差を計算
curve(dnorm(x,mean,sd),-300,150, col = "blue", add = T) # permutation の分布にフィッティングさせた正規分布を描画
```



100 回の permutation では、100 回 × 141 DNA マーカー = 14100 ケースしかないので、p 値は小さくても 7×10^{-5} にしかならない。全遺伝子で同様の検定を行い、多重検定補正を行う場合を想定すると、permutation の回数は数億でも足りない。そこで、確率分布で permutation で得られた残差和の改善度合いの分布を近似して、近似した確率分布に従って p 値を計算する。今回は、正規分布をフィッティングさせている。

```
p.value = pnorm(improvement, mean, sd, lower.tail = F) # p 値を計算。lower.
tail = F : p(x)>improvement
q.value = p.adjust(p.value, method = "fdr") # 多重検定に対する FDR 補正
plot(-log10(q.value), pch = 16) # 補正済み p-value を描画
abline(h=-log10(0.05), col = "red") # 有意水準補正済み p 値<0.05 を描画
```



permutation と確率分布を利用することで、残差和の改善度合いを p 値に変換し、その改善がたまたま起こる確率を調べることができた。

検出された edQTL によって遺伝子発現動態が変化しているのかをチェック

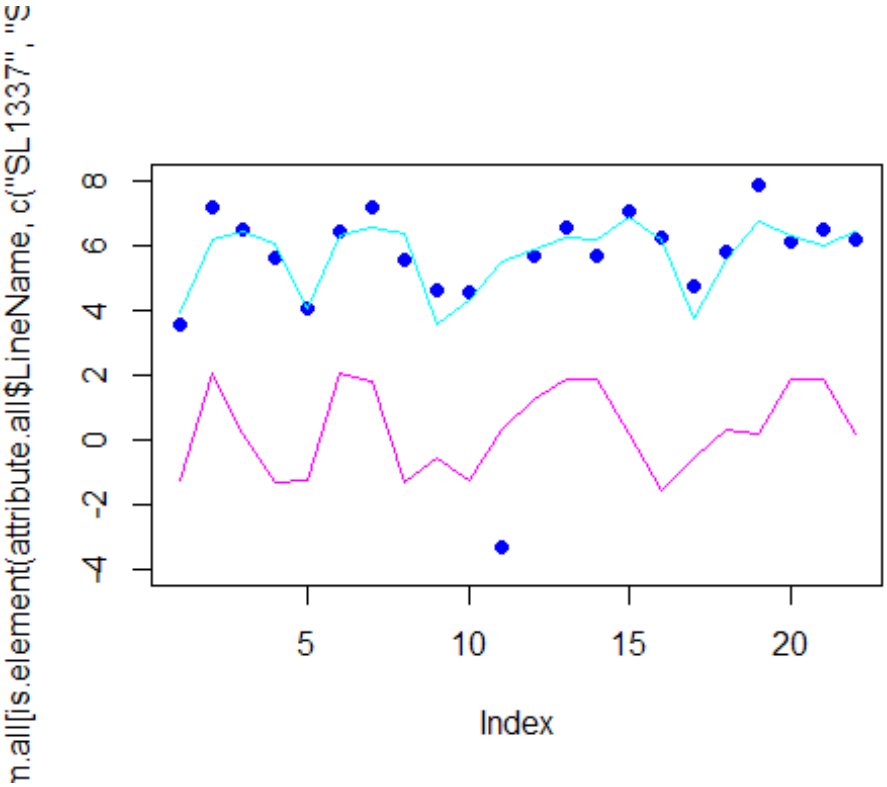
```
edQTL.DNA.markers = which(q.value==min(q.value)) # edQTL を持つマーカーを選択
gt.mat[edQTL.DNA.markers,] # edQTL の genotype を判別
```

```
##     Koshihikari SL1201 SL1202 SL1203 SL1204 SL1205 SL1206 SL1207 SL1208
## 134           A      A      A      A      A      A      A      A      A
## 135           A      A      A      A      A      A      A      A      A
##     SL1209 SL1210 SL1211 SL1212 SL1213 SL1214 SL1215 SL1216 SL1217 SL1218
## 134      A      A      A      A      A      A      A      A      A      A
## 135      A      A      A      A      A      A      A      A      A      A
##     SL1219 SL1220 SL1221 SL1222 SL1223 SL1224 SL1225 SL1226 SL1227 SL1228
## 134      A      A      A      A      A      A      A      A      A      A
## 135      A      A      A      A      A      A      A      A      A      A
##     SL1229 SL1230 SL1231 SL1232 SL1233 SL1234 SL1235 SL1236 SL1237 SL1238
## 134      A      A      A      A      A      A      A      A      A      A
## 135      A      A      A      A      A      A      A      A      A      A
##     SL1239 SL1240 SL1241 SL1301 SL1302 SL1303 SL1304 SL1305 SL1306 SL1307
## 134      A      A      A      B      B      B      B      B      B      B
## 135      A      A      A      B      B      B      B      B      B      B
##     SL1308 SL1309 SL1310 SL1311 SL1312 SL1313 SL1314 SL1315 SL1316 SL1317
## 134      B      B      B      B      B      B      B      B      B      B
## 135      B      B      B      B      B      B      B      B      B      B
##     SL1318 SL1319 SL1320 SL1321 SL1322 SL1323 SL1324 SL1325 SL1326 SL1327
## 134      B      B      B      B      B      B      B      B      B      B
## 135      B      B      B      B      B      B      B      B      B      B
##     SL1328 SL1329 SL1330 SL1331 SL1332 SL1333 SL1334 SL1335 SL1336 SL1337
## 134      B      B      B      B      B      B      B      B      B      A
## 135      B      B      B      B      B      B      B      B      B      A
##     SL1338 SL1339 Takanari HP.a HP.b
## 134      A      B        B    A    B
## 135      A      B        B    A    B
```

対象としている遺伝子を制御している edQTL はコシヒカリ背景の CSSL ではタカナリ型に置換された系統は存在しない。一方で、タカナリ背景の CSSL では、SL1337 と SL1338 で edQTL がコシヒカリ型に置換されている。

```
attribute.tak.BG.substituted = attribute.all[is.element(attribute.all$LineName,c("SL1337","SL1338")),] # edQTL が背景 genotype と異なる系統の attribute を抽出
prediction.attribute  = FIT::convert.attribute(attribute.tak.BG.substituted) # FIT 入力用に attribute.tak.BG.substituted を変換

## # Preparing attribute data..done.
```

```
kos = FIT::predict(models.kos[[gn]], prediction.attribute,train.weather)
# コシヒカリモデルで予測
tak = FIT::predict(models.tak[[gn]], prediction.attribute,train.weather)
# タカナリモデルで予測
plot(log2rpm.all[is.element(attribute.all$LineName,c("SL1337","SL1338"))],
 col = "blue", pch = 16, ylim = c(-4,8)) # 実測値を描画
lines(kos$temperature, col = "cyan") # コシヒカリモデルの予測値を描画
lines(tak$radiation, col = "magenta") # タカナリモデルの予測値を描画
```



SL1337、SL1338 の両方で、実測値の遺伝子発現を、コシヒカリモデルがうまく説明している。一方で、両系統の背景 genotype であるタカナリモデルは外れている。

モデル作成に使用していない genotype の BIL を用いて、edQTL で遺伝子発現の環境応答が説明できるかを検証

最後にモデル作成と edQTL 検出に使していない BIL を用いて、edQTL によって遺伝子発現の環境応答が制御されているのかを検証する。BIL は CSSL と異なり、複数の染色体断片が他系統に置換された、比較的複雑な genotype を持つ。使用する BIL の HP-a と HP-b はタカナリ背景の BIL であり、3/4 の DNA マーカーがタカナリ型である。

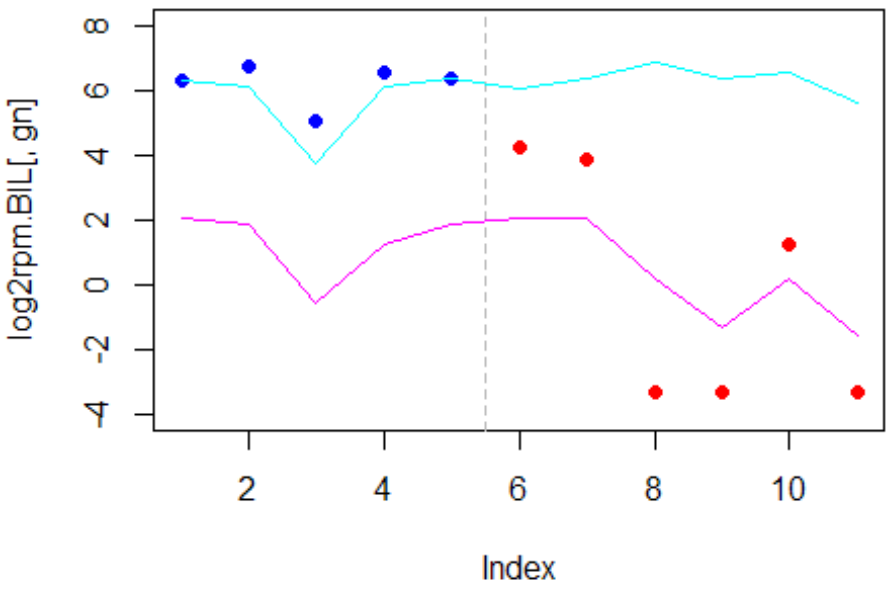
```r
load("BILs_data") # BIL のデータ読み込み
gt.mat[edQTL.DNA.markers,c("HP.a","HP.b")] # 各 BIL の edQTL の genotype を確認
```

```
##     HP.a HP.b
## 134    A    B
## 135    A    B
```

```r
prediction.attribute.BIL  = FIT::convert.attribute(at.BIL) # FIT 入力用に attribute.tak.BG.substituted を変換
```

```
## # Preparing attribute data..done.
```

```r
kos = FIT::predict(models.kos[[gn]], prediction.attribute.BIL,train.weather) # コシヒカリモデルで予測
tak = FIT::predict(models.tak[[gn]], prediction.attribute.BIL,train.weather) # タカナリモデルで予測
plot(log2rpm.BIL[,gn], col = c(rep("blue",sum(at.BIL$LineName=="HP-a")),rep("red",sum(at.BIL$LineName=="HP-b"))), pch = 16, ylim = c(-4,8)) # 実測値を描画
abline(v = sum(at.BIL$LineName=="HP-a")+0.5, col = "gray", lty="dashed") # HP-a と HP-b の境界線の描画
lines(kos$temperature, col = "cyan") # コシヒカリモデルの予測値を描画
lines(tak$radiation, col = "magenta") # タカナリモデルの予測値を描画
```



edQTL の genotype に一致して、対遺伝子の発現は、HP-a ではコシヒカリモデルに、HP-b ではタカナリモデルでよく説明されている。タカナリモデルで予測誤差が大きいのは、前述したタカナリモデルの残差が大きいことと一致している。

結論

本解析を通じて、コシヒカリ・タカナリの遺伝子発現の品種間差を生み出す edQTL を同定することができた。加えて、edQTL の genotype と気象情報から遺伝子発現動態を予測することにも成功している。本解析では、CSSL の利点を利用したコスパの良い実験デザインで実験・解析を行ったが、例えば、親系統を密にサンプリングしてモデルを作成、それから組換え自植系統(recombinant inbred line:RIL)を用いて edQTL 検出するといった実験デザインも可能である。

参考文献

Iwayama, K. et al., 2017. FIT: Statistical modeling tool for transcriptome dynamics under fluctuating field conditions. *Bioinformatics*, 33(11), pp.1672–1680.

Kashima, M. et al., 2018. Prediction of environmental response in field-grown rice using expression-dynamics-QTL. *bioRxiv*, pp.1–38.

Nagano, A.J. et al., 2012. Deciphering and prediction of transcriptome dynamics under fluctuating field conditions. *Cell*, 151(6), pp.1358–1369. Available at: http://dx.doi.org/10.1016/j.cell.2012.10.048.