

Jupyter Notebook für die Analyse von Infinium MethylationEPIC v2.0 BeadChip-Daten in R

Die ChIP Analysis Methylation Pipeline (ChAMP)

Nachdem der Lab-Part der Methylierungsanalyse abgeschlossen ist, erhalten wir als Bioinformatiker*innen Datensätze zusammengesetzt aus einem Samplesheet und mehreren .idat Files, die Fluoreszenzsignale enthalten. Um diese Fluoreszenzsignale (binäres Dateiformat) auslesen und mit dem Samplesheet verknüpfen zu können, greifen wir auf bereits existierende Software zurück. In diesem Fall analysieren wir unsere Daten mit der ChIP Analysis Methylation Pipeline (ChAMP). Diese Pipeline basiert auf R und kombiniert bekannte Software für verschiedene Parts der Methylierungsanalyse (minfi, combat, bumphunter), um eine umfassende Pipeline zur Verfügung zu stellen.

ChAMP unterscheidet bei fast allen Funktionen zwischen drei Typen von Methylierungsarrays, da Illumina über die Jahre verschiedene Arrayversionen veröffentlicht hat. Die älteren Versionen sind 450k- und EPICv1-Arrays, die sich hauptsächlich in der Menge der enthaltenen CpGs unterscheiden. Dieses Notebook ist darauf ausgelegt, die Analyse von EPICv2-Arrays (die aktuellste Version) zu zeigen.

Die hier durchgeführten Schritte umfassen:

1. Einlesen und Filtern der Daten
2. Normalisierung der Methylierungslevel
3. Überprüfung auf und Entfernen von Batch-Effekten
4. Vergleich der Methylierung zwischen Zellproben

Import aller nötigen R Pakete

```
In [1]: # import all needed R packages
library(ChAMP)
library(ChAMPdata)
library(ggplot2)
library(stringr)
library(ggpubr)
```

Lade nötiges Paket: minfi

Lade nötiges Paket: BiocGenerics

Lade nötiges Paket: generics

Attache Paket: 'generics'

Die folgenden Objekte sind maskiert von 'package:base':

as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
setequal, union

Attache Paket: 'BiocGenerics'

Die folgenden Objekte sind maskiert von 'package:stats':

IQR, mad, sd, var, xtabs

Die folgenden Objekte sind maskiert von 'package:base':

anyDuplicated, aperm, append, as.data.frame, basename, cbind,
colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
get, grep, grepl, is.unsorted, lapply, Map, mapply, match, mget,
order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
rbind, Reduce, rownames, sapply, saveRDS, table, tapply, unique,
unsplit, which.max, which.min

Lade nötiges Paket: GenomicRanges

Lade nötiges Paket: stats4

Lade nötiges Paket: S4Vectors

Attache Paket: 'S4Vectors'

Das folgende Objekt ist maskiert 'package:utils':

findMatches

Die folgenden Objekte sind maskiert von 'package:base':

expand.grid, I, unname

Lade nötiges Paket: IRanges

Lade nötiges Paket: GenomeInfoDb

Lade nötiges Paket: SummarizedExperiment

Lade nötiges Paket: MatrixGenerics

Lade nötiges Paket: matrixStats

Attache Paket: 'MatrixGenerics'

Die folgenden Objekte sind maskiert von 'package:matrixStats':

```
colAlls, colAnyNAs, colAnys, colAvgPerRowSet, colCollapse,
colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
colWeightedMeans, colWeightedMedians, colWeightedSds,
colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgPerColSet,
rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
rowWeightedSds, rowWeightedVars
```

Lade nötiges Paket: Biobase

Welcome to Bioconductor

```
Vignettes contain introductory material; view with
'browseVignettes()'. To cite Bioconductor, see
'citation("Biobase")', and for packages 'citation("pkgname")'.
```

Attache Paket: 'Biobase'

Das folgende Objekt ist maskiert 'package:MatrixGenerics':

```
rowMedians
```

Die folgenden Objekte sind maskiert von 'package:matrixStats':

```
anyMissing, rowMedians
```

Lade nötiges Paket: Biostrings

Lade nötiges Paket: XVector

Attache Paket: 'Biostrings'

Das folgende Objekt ist maskiert 'package:base':

strsplit

Lade nötiges Paket: bumphunter

Lade nötiges Paket: foreach

Lade nötiges Paket: iterators

Lade nötiges Paket: parallel

Lade nötiges Paket: locfit

locfit 1.5-9.12 2025-03-05

Setting options('download.file.method.GEOquery'='auto')

Setting options('GEOquery.inmemory.gpl'=FALSE)

Lade nötiges Paket: ChAMPdata

Lade nötiges Paket: DMRcate

Lade nötiges Paket: Illumina450ProbeVariants.db

Lade nötiges Paket: IlluminaHumanMethylationEPICmanifest

Lade nötiges Paket: DT

Lade nötiges Paket: RPMM

Lade nötiges Paket: cluster

Keine Methoden in Paket 'RSQLite' gefunden für Anforderung: 'dbListFields' beim Laden von 'lumi'

Warning message:

"vorhergehender Import 'plyr::mutate' durch 'plotly::mutate' während des Ladens von 'ChAMP' ersetzt"

Warning message:

"vorhergehender Import 'plyr::rename' durch 'plotly::rename' während des Ladens von 'ChAMP' ersetzt"

Warning message:

"vorhergehender Import 'plyr::arrange' durch 'plotly::arrange' während des Ladens von 'ChAMP' ersetzt"

```
Warning message:
"vorhergehender Import 'plyr::summarise' durch 'plotly::summarise' während de
s Ladens von 'ChAMP' ersetzt"

Warning message:
"vorhergehender Import 'plotly::subplot' durch 'Hmisc::subplot' während des L
adens von 'ChAMP' ersetzt"
Warning message:
"vorhergehender Import 'plyr::summarize' durch 'Hmisc::summarize' während des
Ladens von 'ChAMP' ersetzt"
Warning message:
"vorhergehender Import 'plyr::is.discrete' durch 'Hmisc::is.discrete' während
des Ladens von 'ChAMP' ersetzt"
Warning message:
"vorhergehender Import 'plotly::last_plot' durch 'ggplot2::last_plot' während
des Ladens von 'ChAMP' ersetzt"
Warning message:
"vorhergehender Import 'globaltest::model.matrix' durch 'stats::model.matrix'
während des Ladens von 'ChAMP' ersetzt"
Warning message:
"vorhergehender Import 'globaltest::p.adjust' durch 'stats::p.adjust' während
des Ladens von 'ChAMP' ersetzt"
>> Package version 2.29.1 loaded <<
```



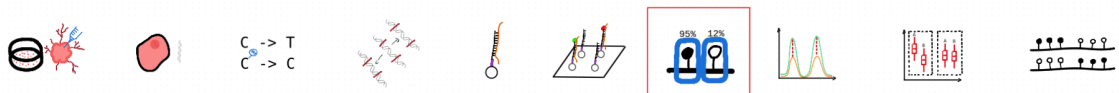
If you have any question or suggestion about ChAMP, please email to champ450k@gmail.com.

Thank you for citing ChAMP:

Yuan Tian, Tiffany J Morris, Amy P Webster, Zhen Yang, Stephan Beck, Andrew Feber, Andrew E Teschendorff; ChAMP: updated methylation analysis pipeline for Illumina BeadChips, Bioinformatics, btx513, <https://doi.org/10.1093/bioinformatics/btx513> REQUIRE ChAMPdata >= 2.23.1

```
In [2]: # set the location of the experiment directory
# the .idat fluorescence files and the samplesheet.csv have to be located in
epicv2_dir <- "../..example_datasets/epicv2/data"
```

Setzen des Datenverzeichnis, Laden und Filtern der Daten



Alle Daten müssen für die Verarbeitung mit ChAMP in einem Verzeichnis abgelegt werden. Das Verzeichnis enthält sowohl alle .idat Dateien (jeweils grün und rot pro Probe), als auch das Samplesheet. ChAMP sucht initial eine .csv Datei, die als Samplesheet gelesen wird (bei mehr als einer .csv Dateien treten Probleme auf), liest daraus Sentrix_ID und

Sentrix_Position, und sucht anhand dieser Informationen die .idat Dateien benannt in der Form *Sentrix_ID*Sentrix_position*_Grn.idat* und **Sentrix_ID*\Sentrix_position_Red.idat*.

Die Fluoreszenzsignale werden daraufhin ausgelesen und daraus je nach benötigtem Methylierungswert (**methValue**) Beta oder M Werte berechnet. Zusätzlich wird eine Reihe von Filtern angewendet, die folgende Sonden aus den Datensätzen entfernen:

1. Sonden, die nicht signifikant methyliert oder demethyliert sind (die sich nicht signifikant vom Hintergrundsignal unterscheiden)
2. Sonden, die nicht die Methylierung von CpGs überprüfen
3. Sonden, die in der Nähe von single nucleotide polymorphisms (SNPs) liegen
4. Sonden, die auf mehrere genomische Regionen mappen
5. Sonden, die auf dem X- und Y-Chromosom liegen
6. Sonden, in denen der Anteil der Proben mit weniger als 3 Beads über **beadCutoff** liegt

Zusätzlich werden Proben entfernt, in denen mindestens **SampleCutoff** Sonden durch vorherige Filterungsschritte entfernt wurden. Alle **fett** gedruckten Begriffe sind Argumente der unten stehenden `champ.load` Funktion

Der Stdout und Stderr enthält von ChAMP sehr viele nützliche Informationen zu Datensatz und Filterprozess. So kann man beispielsweise sehen, wie viele Sonden in den verschiedenen Filterungsschritten entfernt wurden und, ob Proben entfernt wurden (abhängig von **SampleCutoff**).

```
In [3]: # champ.load searches for a samplesheet file and all .idat files listed in s
myLoad <- champ.load(directory = epicv2_dir,
                      method="ChAMP", # replaces old loading method using mir
                      methValue="B", # wether to calculate beta- or M-values
                      autoimpute=TRUE, # if values are missing uses the 3 mos
                      filterDetP=TRUE, # filter single probes, whose methylat
                      ProbeCutoff=0, # remove all probes with higher missing-
                      SampleCutoff=0.1, # remove a full sample if the failed
                      detPcut=0.01, # significance p-value cutoff for filterD
                      filterBeads=TRUE, # filter out probes if the fraction c
                      beadCutoff=0.05, # acceptable fraction of samples with
                      filterNoCG=TRUE, # wether to remove non-cg probes
                      filterSNPs=TRUE, # wether to remove probes that fallnea
                      population=NULL, # can be assigned to specific populati
                      filterMultiHit=TRUE, # wether to remove probes that ali
                      filterXY=TRUE, # wether to remove probes on x and y chr
                      force=FALSE, # minfi specific parameter
                      arraytype="EPICv2") # microarray type (can be one of "4
```

```
[=====]
```

```
[<<<< ChAMP.LOAD START >>>>]
```

```
-----
```

```
[ Loading Data with ChAMP Method ]
```

```
-----
```

Note that ChAMP method will NOT return rgSet or mset, they object defined by minfi. Which means, if you use ChAMP method to load data, you can not use SWAN or FunctionNormlization method in champ.norm() (you can use BMIQ or PBC still). But All other function should not be influenced.

```
[=====]
```

```
[<<<< ChAMP.IMPORT START >>>>]
```

```
-----
```

```
[ Section 1: Read PD Files Start ]
```

CSV Directory: ../../example_datasets/epicv2/data/Demo_EPIC-8v2-0_A1_Sample Sheet_16.csv

Find CSV Success

Reading CSV File

Replace Sentrix_Position into Array

Replace Sentrix_ID into Slide

```
[ Section 1: Read PD file Done ]
```

```
[ Section 2: Read IDAT files Start ]
```

Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R01C01_Grn.idat ---- (1/16)

Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R03C01_Grn.idat ---- (2/16)

Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R07C01_Grn.idat ---- (3/16)

Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R08C01_Grn.idat ---- (4/16)

Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R02C01

```
_Grn.idat ---- (5/16)

  Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R04C01
_Grn.idat ---- (6/16)

  Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R06C01
_Grn.idat ---- (7/16)

  Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R08C01
_Grn.idat ---- (8/16)

  Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R01C01
_Grn.idat ---- (9/16)

  Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R03C01
_Grn.idat ---- (10/16)

  Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R07C01
_Grn.idat ---- (11/16)

  Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R08C01
_Grn.idat ---- (12/16)

  Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R02C01
_Grn.idat ---- (13/16)

  Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R04C01
_Grn.idat ---- (14/16)

  Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R06C01
_Grn.idat ---- (15/16)

  Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R08C01
_Grn.idat ---- (16/16)

  Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R01C01
_Red.idat ---- (1/16)

  Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R03C01
_Red.idat ---- (2/16)

  Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R07C01
_Red.idat ---- (3/16)

  Loading:../../example_datasets/epicv2/data/206891110001/206891110001_R08C01
_Red.idat ---- (4/16)

  Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R02C01
_Red.idat ---- (5/16)

  Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R04C01
_Red.idat ---- (6/16)

  Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R06C01
_Red.idat ---- (7/16)
```



```
Loading:../../example_datasets/epicv2/data/206891110002/206891110002_R08C01_Red.idat ---- (8/16)

Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R01C01_Red.idat ---- (9/16)

Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R03C01_Red.idat ---- (10/16)

Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R07C01_Red.idat ---- (11/16)

Loading:../../example_datasets/epicv2/data/206891110004/206891110004_R08C01_Red.idat ---- (12/16)

Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R02C01_Red.idat ---- (13/16)

Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R04C01_Red.idat ---- (14/16)

Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R06C01_Red.idat ---- (15/16)

Loading:../../example_datasets/epicv2/data/206891110005/206891110005_R08C01_Red.idat ---- (16/16)
```

Extract Mean value for Green and Red Channel Success

Your Red Green Channel contains 1105209 probes.

[Section 2: Read IDAT Files Done]

[Section 3: Use Annotation Start]

Reading EPICv2 Annotation >>

!!! Important, since version 2.29.1, ChAMP set default `EPIC` arraytype as EPIC version 2.

You can set 'EPIC' or 'EPICv2' to use version 2 EPIC annotation

If you want to use the old version (v1), please specify arraytype parameter as `EPICv1`.

For 450K array, still use `450K`

Fetching NEGATIVE ControlProbe.

Totally, there are 411 control probes in Annotation.

Your data set contains 411 control probes.

Generating Meth and UnMeth Matrix

Extracting Meth Matrix...

Totally there are 937055 Meth probes in EPICv2 Annotation.

Your data set contains 937055 Meth probes.

Extracting UnMeth Matrix...

Totally there are 937055 UnMeth probes in EPICv2 Annotation.

Your data set contains 937055 UnMeth probes.

Generating beta Matrix

Generating M Matrix

Generating intensity Matrix

Calculating Detect P value

Counting Beads

[Section 3: Use Annotation Done]

[<<<<< ChAMP.IMPORT END >>>>>]

[=====]

[You may want to process champ.filter() next.]

[=====]

[<<<< ChAMP.FILTER START >>>>]

In New version ChAMP, champ.filter() function has been set to do filtering on the result of champ.import(). You can use champ.import() + champ.filter() to do Data Loading, or set "method" parameter in champ.load() as "ChAMP" to get the same effect.

This function is provided for user need to do filtering on some beta (or M) matrix, which contained most filtering system in champ.load except beadcount. User need to input beta matrix, pd file themselves. If you want to do filtering on detP matrix and Bead Count, you also need to input a detected P matrix and Bead Count information.

Note that if you want to filter more data matrix, say beta, M, intensity... p

lease make sure they have exactly the same rownames and colnames.

[Section 1: Check Input Start]

You have inputed beta,intensity for Analysis.

pd file provided, checking if it's in accord with Data Matrix...

pd file check success.

Parameter filterDetP is TRUE, checking if detP in accord with Data Matrix...

detP check success.

Parameter filterBeads is TRUE, checking if beadcount in accord with Data Matrix...

beadcount check success.

parameter autoimpute is TRUE. Checking if the conditions are fulfilled...

!!! ProbeCutoff is 0, which means you have no needs to do imputation. autoimpute has been reset FALSE.

Checking Finished :filterDetP,filterBeads,filterMultiHit,filterSNPs,filterNoCG,filterXY would be done on beta,intensity.

You also provided :detP,beadcount .

[Section 1: Check Input Done]

[Section 2: Filtering Start >>

Filtering Detect P value Start

The fraction of failed positions per sample

You may need to delete samples with high proportion of failed probes:

Failed CpG Fraction.

HeLa	0.008623827
Jurkat	0.008777500
Raji	0.010284348
NA12873	0.007755148
NA12873_2_R	0.005907871
Raji_2_R	0.008498968
HeLa_2_R	0.009623768
Jurkat_2_R	0.013604324
HeLa_3_R	0.008403989
Jurkat_3_R	0.007717797
Raji_3_R	0.012234074
NA12873_3_R	0.009718746
NA12873_4_R	0.006075417
Raji_4_R	0.008782836
HeLa_4_R	0.010018622
Jurkat_4_R	0.014514623

Filtering probes with a detection p-value above 0.01.

Removing 30127 probes.

If a large number of probes have been removed, ChAMP suggests you to identify potentially bad samples

Filtering BeadCount Start

Filtering probes with a beadcount <3 in at least 5% of samples.

Removing 20547 probes

Filtering NoCG Start

Only Keep CpGs, removing 3445 probes from the analysis.

Filtering SNPs Start

!!! Important, since version 2.29.1, ChAMP set default `EPIC` arraytype as EPIC version 2.

You can set 'EPIC' or 'EPICv2' to use version 2 EPIC annotation

If you want to use the old version (v1), please specify arraytype parameter as `EPICv1`.

For 450K array, still use `450K`

Using general mask options

Removing 30398 probes from the analysis.

Filtering MultiHit Start

Filtering probes that align to multiple locations as identified in Nordlund et al

Removing 0 probes from the analysis.

Filtering XY Start

Filtering probes located on X,Y chromosome, removing 19986 probes from the analysis.

Updating PD file

Fixing Outliers Start

Replacing all value smaller/equal to 0 with smallest positive value.

Replacing all value greater/equal to 1 with largest value below 1..

[Section 2: Filtering Done]

All filterings are Done, now you have 832552 probes and 16 samples.

[<<<<< ChAMP.FILTER END >>>>>]

[=====]

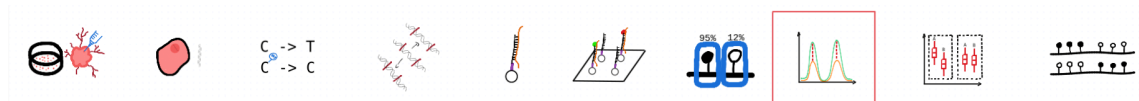
[You may want to process champ.QC() next.]

[<<<<< ChAMP.LOAD END >>>>>]

[=====]

[You may want to process champ.QC() next.]

Qualitätskontrolle des Datensatzes vor der Normalisierung



Das Objekt *MyLoad* enthält, je nach *methValue* Beta- oder M-Werte für alle Sonden und Proben, die nach der Filterung übrig sind. Nach dem Filterungsprozess sollten die Daten visualisiert werden, um eine erste Qualitätskontrolle durchführen zu können.

`champ.QC` erstellt uns aus dem Datensatz einen MDS-Plot, einen Density-Plot von Beta- oder M-Werten und ein Dendrogramm der Proben.

MDS-Plot: Ähnlich einer Principal component analysis (PCA) dient der MDS-Plot dazu, die Daten pro Probe in einen niedrigdimensionalen Raum zu projizieren. Für jede Probe bleiben zwei Werte übrig, die die Unterschiede zwischen den Proben bestmöglich beschreiben. Im Idealfall sollten Replikate sehr stark Clustern und die Samplegroups sollten in den meisten Fällen gut separiert sein.

Density-Plot: Da die DNA, die wir auf die Microarrays geben aus einer Population von Zellen (keiner Einzelzelle) stammt, zeigt uns die Fluoreszenz die durchschnittliche Methylierung einer Menge von Zellen. Der Density-Plot zeigt die Dichteverteilung der Beta-Werte (Methylierung) pro Probe. Die hier dargestellten Kurven sollten eine Badewannen-Form haben: Zwei klare Peaks, der eine um 0, der andere um 1. Zwischen den beiden Peaks sollte die Kurve relativ flach sein. Diese Form ist zu erwarten, da wir annehmen, dass

die Methylierung der meisten CpGs in einer Probe über mehrere Zellen gleich sein sollte - entweder methyliert oder unmethyliert.

Dendrogramm: Das Dendrogramm hat eine ähnliche Funktion, wie der MDS-Plot. Hier werden alle Proben ihrer Ähnlichkeit entsprechend aufgetrennt. Die Länge des Weges zwischen zwei Proben spiegelt die Ähnlichkeit der Proben wieder. Hier ist zu erwarten, dass eine klare Separation der Proben zwischen den Samplegroups und Replikate stattfindet.

In allen Plots kann schon jetzt überprüft werden, ob die Samplegroups sich in Dendrogramm und MDS-Plot gut separieren. Das muss nicht immer der Fall sein, abhängig von den zu erwartenden Unterschieden in der Methylierung zwischen Samplegroups.

Wenn die Auftrennung der Proben in Samplegroups unerwartet schwach ist, kann dies darauf hindeuten, dass zwischen den Proben nur kleine Unterschiede in der Methylierung vorliegen. Wenn die Auftrennung der Replikate unerwartet groß sein sollte, kann dies darauf hindeuten, dass es im experimentellen Ablauf zu Batch-Effekten kam (mehr Info hierzu im Kapitel **Erkennen und Entfernen von Batch Effekten**). Um das genauer zu betrachten, könnt ihr pheno anpassen, um Auftrennungen nach anderen Spalten eures Samplesheets zu betrachten.

```
In [4]: champ.QC(beta = myLoad$beta, # beta values stored in champ.load output
               pheno=myLoad$pd$Sample_Group, # what samplesheet column is your phe
               resultsDir="./CHAMP_QCimages/") # the plots will be saved in the di
```

```
[=====]
[<<<<< ChAMP.QC START >>>>>]
-----

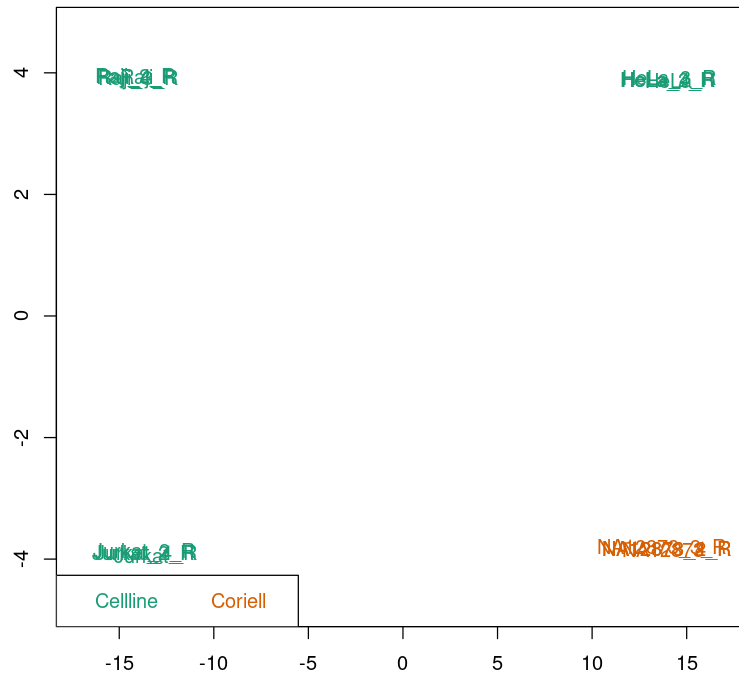
champ.QC Results will be saved in ./CHAMP_QCimages/

[QC plots will be proceed with 832552 probes and 16 samples.]

<< Prepare Data Over. >>

<< plot mdsPlot Done. >>
```

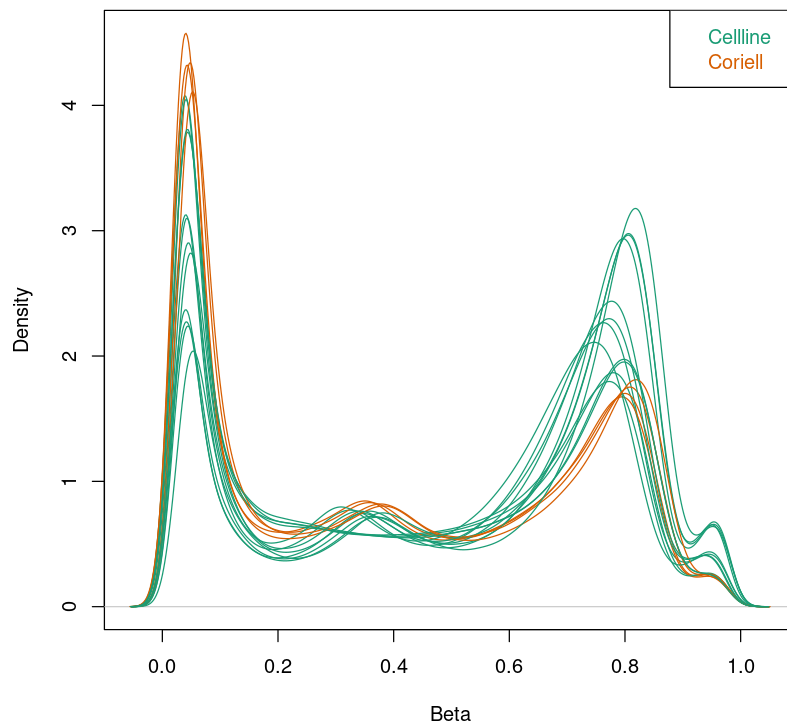
Beta MDS
1000 most variable positions



```
<< Plot densityPlot Done. >>
```

< Dendrogram Plot Feature Selection Method >: No Selection, directly use all CpGs to calculate distance matrix.

Density plot of raw data (832552 probes)

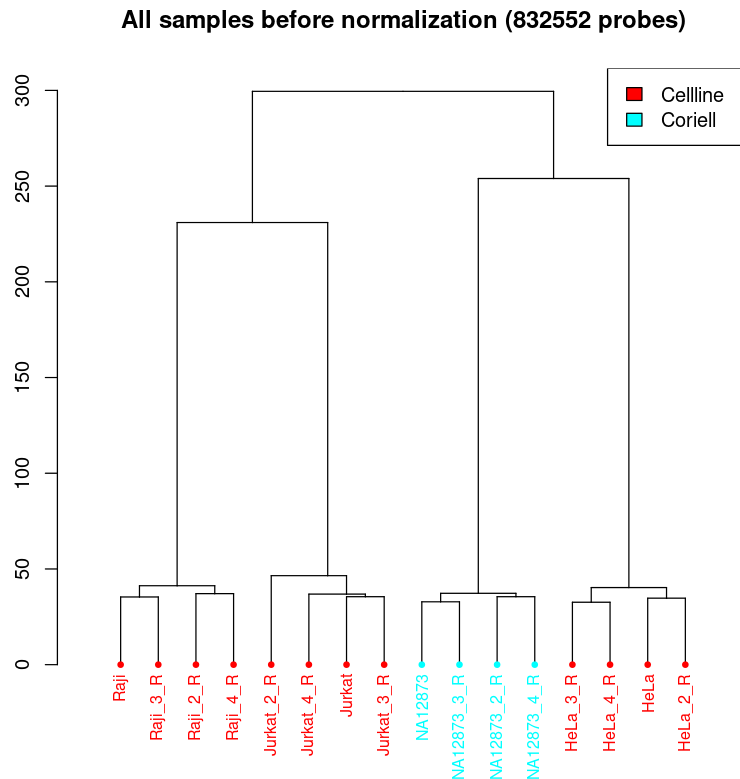



```
<< Plot dendrogram Done. >>
```

```
[<<<<< ChAMP.QC END >>>>>]
```

```
[=====]
```

```
[You may want to process champ.norm() next.]
```



Normalisierung mit anschließender Evaluation

Nachdem wir die Daten im oberen Abschnitt eingelesen und gefiltert haben, werden diese normalisiert, um die Vergleichbarkeit der Daten zu verbessern, ohne sie zu verfälschen. Dafür vergleichen wir den Effekt der Normalisierung durch PBC und BMIQ (unterschiedliche Normalisierungsmethoden). ChAMP bietet zusätzlich funktionale Normalisierung und SWAN, diese beruhen allerdings auf `rgset` und `mset`, weshalb wir sie an dieser Stelle nicht nutzen können.

Nach jeder Normalisierung nutzen wir erneut `champ.QC`, um eine Qualitätskontrolle durchzuführen. Am Ende entscheiden wir uns für die Normalisierung mit der besten Separation und der besten Badewanne.

```
In [5]: # additional data needed for normalization
targets <- read.metharray.sheet(epicv2_dir)
```

```
rgset <- read.metharray.exp(targets = targets, recursive = TRUE)
mset <- preprocessRaw(rgset)
```

[read.metharray.sheet] Found the following CSV files:

```
[1] "../..example_datasets/epicv2/data/Demo_EPIC-8v2-0_A1_SampleSheet_16.csv"
```

Lade nötiges Paket: IlluminaHumanMethylationEPICv2manifest

Peak-based correction normalization (PBC)

```
In [6]: # start with PBC normalization
myNormPBC <- champ.norm(beta = myLoad$beta, arraytype = "EPICv2", method = "
champ.QC(beta = myNormPBC, pheno = myLoad$pd$Sample_Group, resultsDir = ".")

[=====]

[>>>> ChAMP.NORM START <<<<<]

-----

champ.norm Results will be saved in ./CHAMP_Normalization/

[ SWAN method call for BOTH rgSet and mset input, FunctionalNormalization call
for rgset only , while PBC and BMIQ only needs beta value. Please set parameter
correctly. ]

<< Normalizing data with PBC Method >>

[1] "Done for sample 1"
[1] "Done for sample 2"
[1] "Done for sample 3"
[1] "Done for sample 4"
[1] "Done for sample 5"
[1] "Done for sample 6"
[1] "Done for sample 7"
[1] "Done for sample 8"
[1] "Done for sample 9"
[1] "Done for sample 10"
[1] "Done for sample 11"
[1] "Done for sample 12"
[1] "Done for sample 13"
[1] "Done for sample 14"
[1] "Done for sample 15"
[1] "Done for sample 16"
```

```
[>>>> ChAMP.NORM END <<<<<]

[=====]

[You may want to process champ.SVD() next.]

[=====]

[<<<<< ChAMP.QC START >>>>>]

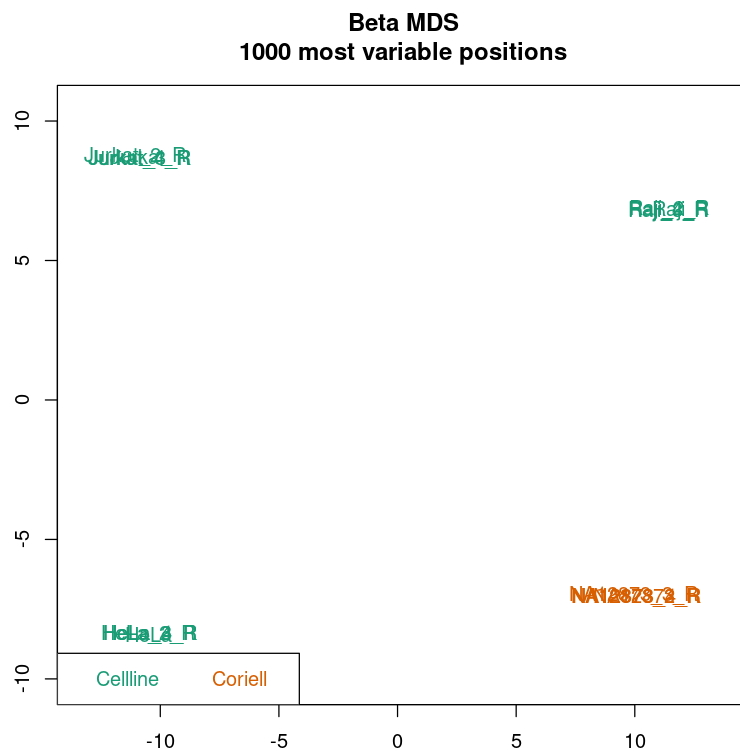
-----

champ.QC Results will be saved in ./CHAMP_QCimages/PBC/

[QC plots will be proceed with 832552 probes and 16 samples.]

<< Prepare Data Over. >>

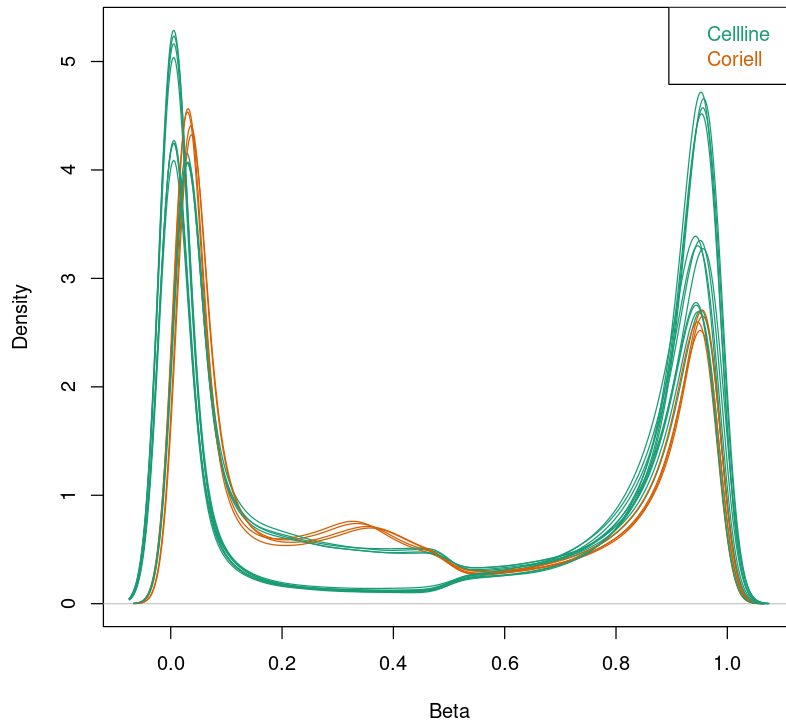
<< plot mdsPlot Done. >>
```



```
<< Plot densityPlot Done. >>
```

```
< Dendrogram Plot Feature Selection Method >: No Selection, directly use all
CpGs to calculate distance matrix.
```

Density plot of raw data (832552 probes)



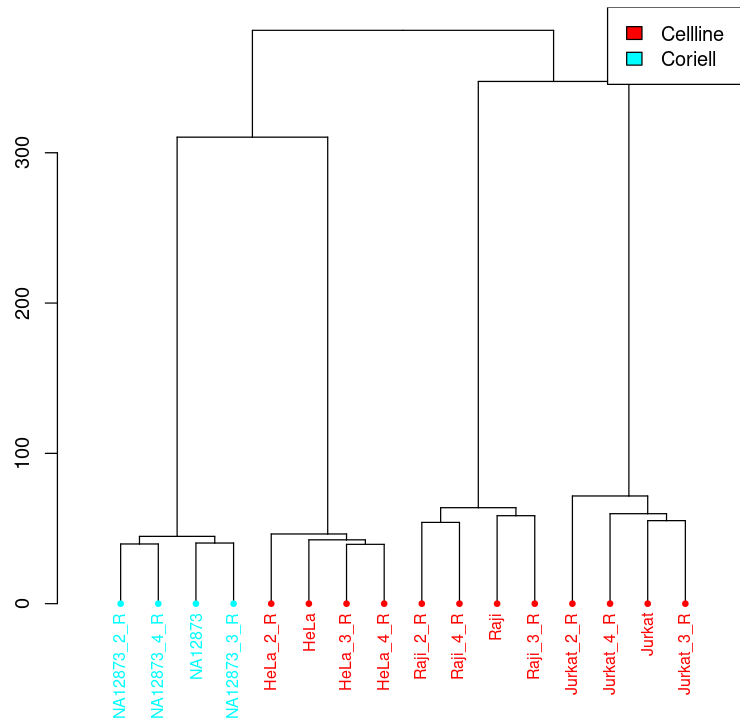
```
<< Plot dendrogram Done. >>
```

```
[<<<<< ChAMP.QC END >>>>>]
```

```
[=====]
```

```
[You may want to process champ.norm() next.]
```

All samples before normalization (832552 probes)



Beta-mixture quantile normalization (BMIQ)

```
In [7]: # BMIQ normalization
myNormBMIQ <- champ.norm(beta = myLoad$beta, arraytype = "EPICv2", method =
champ.QC(beta = myNormBMIQ, pheno=myLoad$pd$Sample_Group, resultsDir="./CHAMP
```

```

[=====]

[>>>> ChAMP.NORM START <<<<<]

-----

champ.norm Results will be saved in ./CHAMP_Normalization/

[ SWAN method call for BOTH rgSet and mset input, FunctionalNormalization call for rgset only , while PBC and BMIQ only needs beta value. Please set parameter correctly. ]

<< Normalizing data with BMIQ Method >>

Note that,BMIQ function may fail for bad quality samples (Samples did not even show beta distribution).

3 cores will be used to do parallel BMIQ computing.

[>>>> ChAMP.NORM END <<<<<]

[=====]

[You may want to process champ.SVD() next.]

[=====]

[<<<<< ChAMP.QC START >>>>>]

-----

champ.QC Results will be saved in ./CHAMP_QCimages/BMIQ/

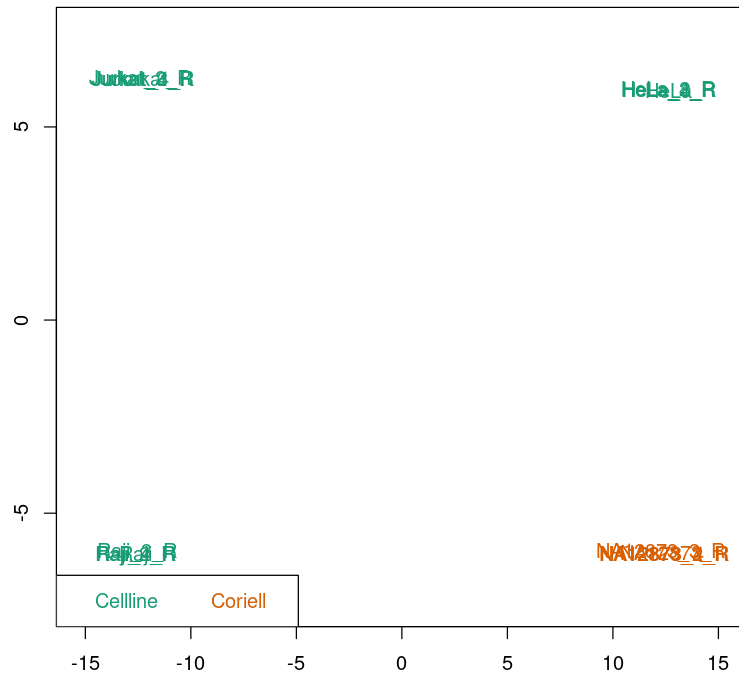
[QC plots will be proceed with 832552 probes and 16 samples.]

<< Prepare Data Over. >>

<< plot mdsPlot Done. >>

```

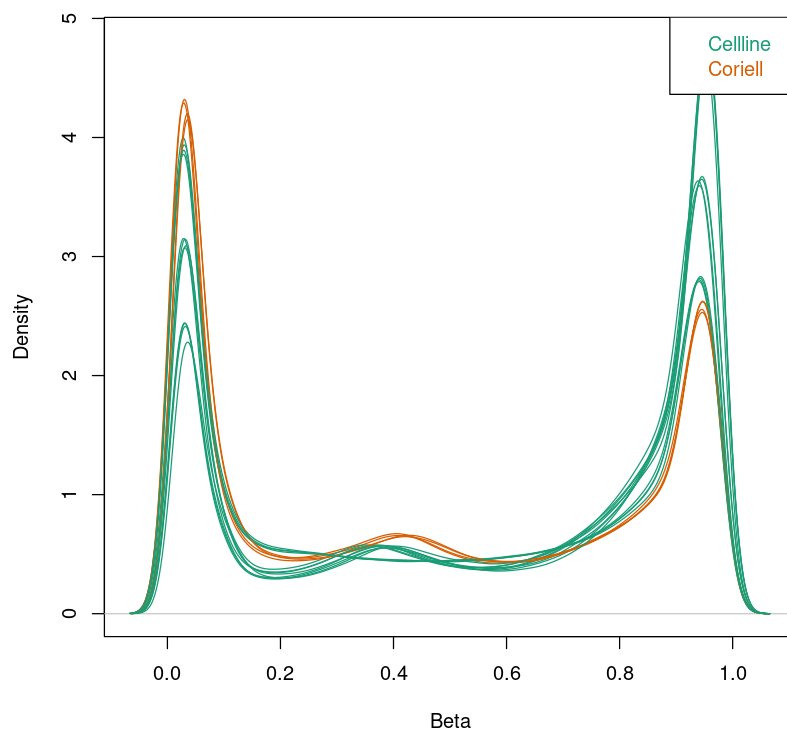
Beta MDS
1000 most variable positions



```
<< Plot densityPlot Done. >>
```

< Dendrogram Plot Feature Selection Method >: No Selection, directly use all CpGs to calculate distance matrix.

Density plot of raw data (832552 probes)



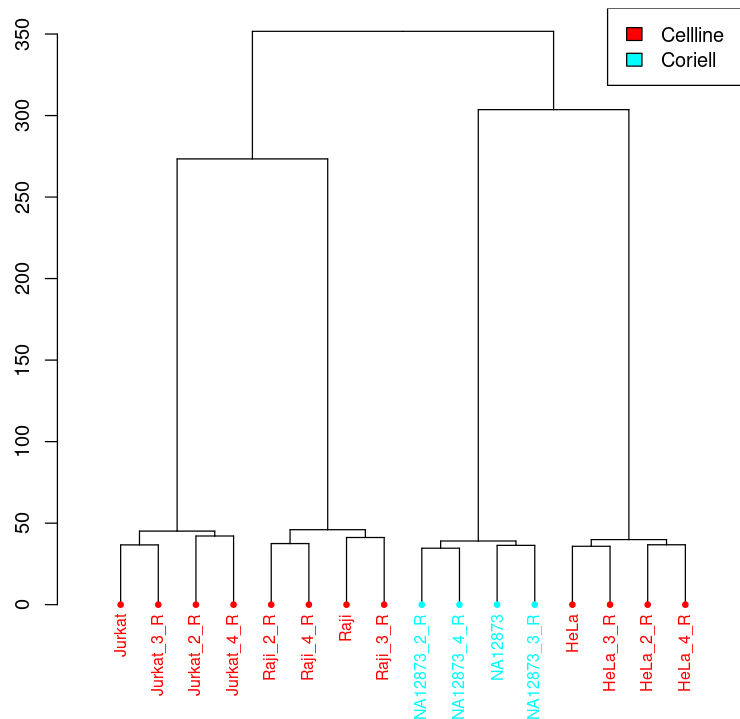
```
<< Plot dendrogram Done. >>
```

```
[<<<<< ChAMP.QC END >>>>>]
```

```
[=====]
```

```
[You may want to process champ.norm() next.]
```

All samples before normalization (832552 probes)



Subset-quantiles within microarray normalization (SWAN)

```
In [8]: # SWAN normalization (no native ChAMP support)
myNormSWAN <- champ.norm(arraytype = "EPICv2", method = "SWAN", rgSet=rgset,
colnames(myNormSWAN) <- targets$Sample_Name
pheno <- targets$Sample_Group
names(pheno) <- targets$Sample_Name # Echte Namen
pheno <- pheno[colnames(myNormSWAN)] # Reihenfolge anpassen
champ.QC(beta = myNormSWAN, pheno = myLoad$pd$Sample_Group, resultsDir = ".")
```



```
[=====]

[>>>> ChAMP.NORM START <<<<<]

-----

champ.norm Results will be saved in ./CHAMP_Normalization/

[ SWAN method call for BOTH rgSet and mset input, FunctionalNormalization call for rgset only , while PBC and BMIQ only needs beta value. Please set parameter correctly. ]

[>>>> ChAMP.NORM END <<<<<]

[=====]

[You may want to process champ.SVD() next.]

[=====]

[<<<<< ChAMP.QC START >>>>>]

-----

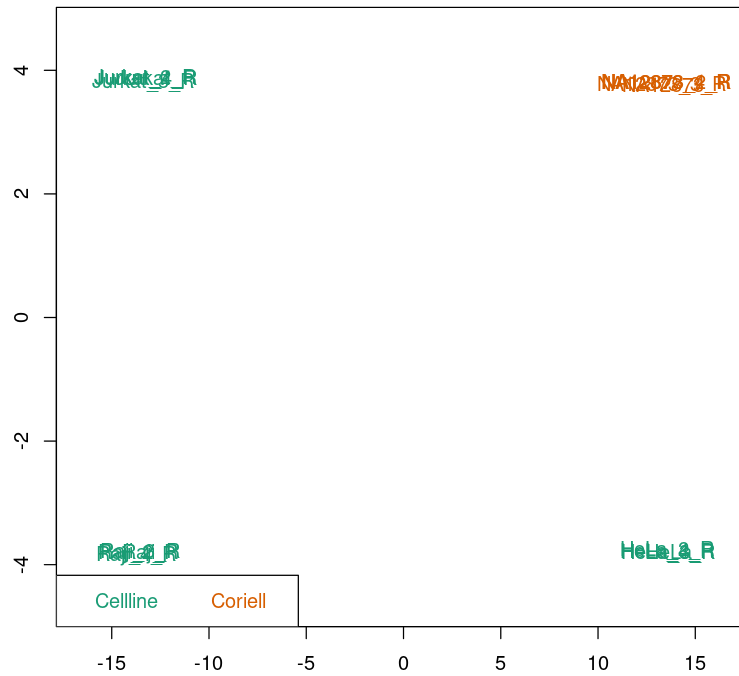
champ.QC Results will be saved in ./CHAMP_QCimages/SWAN/

[QC plots will be proceed with 936990 probes and 16 samples.]

<< Prepare Data Over. >>

<< plot mdsPlot Done. >>
```

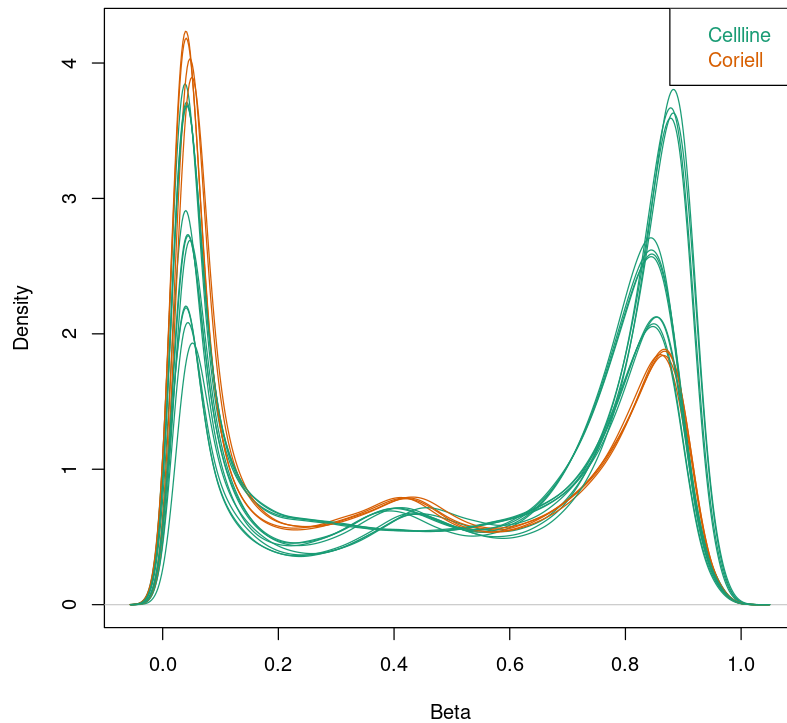
Beta MDS
1000 most variable positions



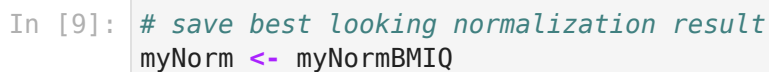
```
<< Plot densityPlot Done. >>
```

< Dendrogram Plot Feature Selection Method >: No Selection, directly use all CpGs to calculate distance matrix.

Density plot of raw data (936990 probes)



[You may want to process `champ.norm()` next.]



In solchen Fällen kann die Ursache bei Batch-Effekten liegen. Als Batch-Effekt bezeichnet man (nicht zwingend erklärbare) Unterschiede in der durchschnittlichen Fluoreszenz

zwischen verschiedenen Beadchips, Positionen auf dem Beadchip oder beispielsweise Tagen, an denen Experimente durchgeführt wurden.

Um diese Effekte zu finden, wird das Samplesheet und alle darin stehenden Gruppierungsspalten genutzt, um nach Unterschieden zwischen Gruppen zu suchen. Zu diesen Gruppen gehören Sample_Group, Satrix_ID, Satrix_Position und Pool_ID (falls ausgefüllt).

`champ.SVD` erstellt einen SVD Plot, in dem erkennbar ist, wie stark verschiedene Gruppierungsparameter mit Varianz zwischen den Proben (wiedergespiegelt durch die ersten drei Hauptkomponenten PC-1, PC-2 und PC-3) assoziiert sind. Alle bunten Blöcke sind dabei statistisch signifikant ($p < 0.05$) und sollten berücksichtigt werden mit Ausnahme der Blöcke für Samplegroup.

Nach Feststellung von Batch-Effekten kann `champ.runCombat` genutzt werden, um die gefundenen Batch-Effekte zu entfernen.

Zur Auswertung dient ein weiterer SVD-Plot.

```
In [10]: champ.SVD(beta=myNorm, pd=myLoad$pd)
```

```
[=====]
```

```
[<<<<< ChAMP.SVD START >>>>>]
```

```
-----
```

champ.SVD Results will be saved in ./CHAMP_SVDimages/ .

[SVD analysis will be proceed with 832552 probes and 16 samples.]

[champ.SVD() will only check the dimensions between data and pd, instead if checking if Sample_Names are correctly matched (because some user may have no Sample_Names in their pd file),thus please make sure your pd file is in accord with your data sets (beta) and (rgSet).]

<< Following Factors in your pd(sample_sheet.csv) will be analysed: >>

<Sample_Group>(character):Cellline, Coriell

<Slide>(character):206891110001, 206891110002, 206891110004, 206891110005

<Array>(character):R01C01, R03C01, R07C01, R08C01, R02C01, R04C01, R06C01

[champ.SVD have automatically select ALL factors contain at least two different values from your pd(sample_sheet.csv), if you don't want to analysis some of them, please remove them manually from your pd variable then retry champ.SVD().]

<< Following Factors in your pd(sample_sheet.csv) will not be analysis: >>

<Sample_Name>

<Sample_Well>

<Sample_Plate>

<Pool_ID>

[Factors are ignored because they only indicate Name or Project, or they contain ONLY ONE value across all Samples.]

<< PhenoTypes.lv generated successfully. >>

<< Calculate SVD matrix successfully. >>

<< Plot SVD matrix successfully. >>

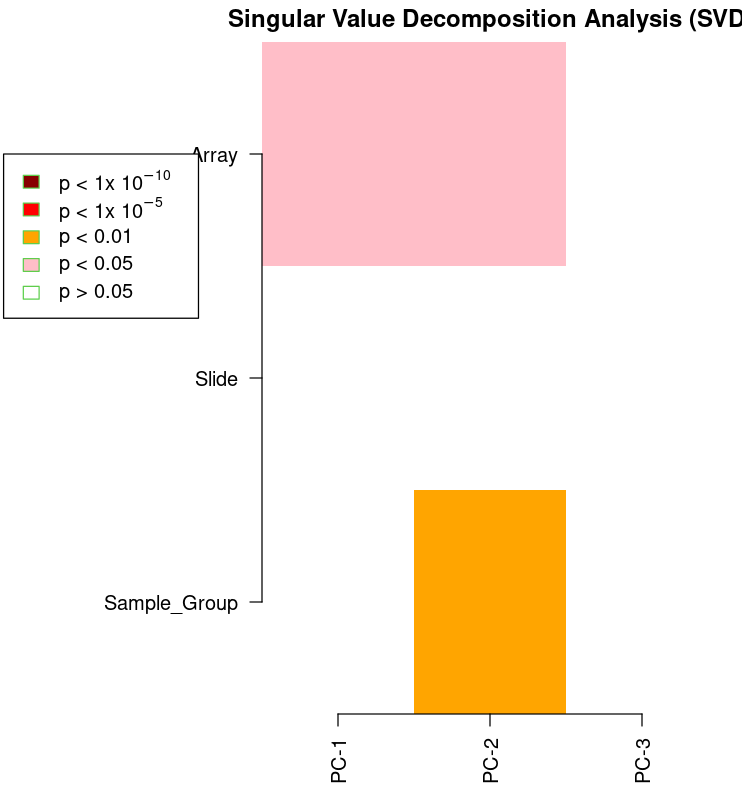
```
[<<<<< ChAMP.SVD END >>>>>]
```

```
[=====]
```

[If the batch effect is not significant, you may want to process `champ.DMP()` or `champ.DMR()` or `champ.BlockFinder()` next, otherwise, you may want to run `champ.runCombat()` to eliminat batch effect, then rerun `champ.SVD()` to check cor rected result.]

A matrix: 3 × 3 of type dbl

Sample_Group	Slide	Array
0.331975467	0.9624982	0.02799695
0.003609347	0.8974416	0.02607784
0.331975467	0.9541195	0.07487659



```
In [11]: myCombat <- champ.runCombat(beta=myNorm,pd=myLoad$pd,batchname=c("Array"))
```

```
[=====]
[<< CHAMP.RUNCOMBAT START >>]
-----

<< Preparing files for ComBat >>

[Combat correction will be proceed with 832552 probes and 16 samples.]

<< Following Factors in your pd(sample_sheet.csv) could be applied to Combat:
>>

<Sample_Name>(character)
<Slide>(character)
<Array>(character)

[champ.runCombat have automatically select ALL factors contain at least two d
ifferent values from your pd(sample_sheet.csv).]

<< Following Factors in your pd(sample_sheet.csv) can not be corrected: >>

<Sample_Well>
<Sample_Plate>
<Sample_Group>
<Pool_ID>

[Factors are ignored because they are conflict with variablename, or they con
tain ONLY ONE value across all Samples, or some phenotype contains less than
2 Samples.]

As your assigned in batchname parameter: Array will be corrected by Combat fu
nction.

<< Start Correcting Array >>

~Sample_Group
<environment: 0x61a34ce500f0>

Generate mod success. Started to run ComBat, which is quite slow...

Found 4 genes with uniform expression within a single batch (all zeros); thes
e will not be adjusted for batch.
```

Found 7 batches

Adjusting for 1 covariate(s) or covariate level(s)

Standardizing Data across genes

Fitting L/S model and finding priors

Finding parametric adjustments

Adjusting the Data

champ.runCombat success. Corrected dataset will be returned.

```
In [12]: champ.SVD(beta=myCombat, pd=myLoad$pd)
```



```
[=====]
```

```
[<<<<< ChAMP.SVD START >>>>>]
```

```
-----
```

champ.SVD Results will be saved in ./CHAMP_SVDimages/ .

[SVD analysis will be proceed with 832552 probes and 16 samples.]

[champ.SVD() will only check the dimensions between data and pd, instead if checking if Sample_Names are correctly matched (because some user may have no Sample_Names in their pd file),thus please make sure your pd file is in accord with your data sets (beta) and (rgSet).]

<< Following Factors in your pd(sample_sheet.csv) will be analysed: >>

<Sample_Group>(character):Cellline, Coriell

<Slide>(character):206891110001, 206891110002, 206891110004, 206891110005

<Array>(character):R01C01, R03C01, R07C01, R08C01, R02C01, R04C01, R06C01

[champ.SVD have automatically select ALL factors contain at least two different values from your pd(sample_sheet.csv), if you don't want to analysis some of them, please remove them manually from your pd variable then retry champ.SVD().]

<< Following Factors in your pd(sample_sheet.csv) will not be analysis: >>

<Sample_Name>

<Sample_Well>

<Sample_Plate>

<Pool_ID>

[Factors are ignored because they only indicate Name or Project, or they contain ONLY ONE value across all Samples.]

<< PhenoTypes.lv generated successfully. >>

<< Calculate SVD matrix successfully. >>

<< Plot SVD matrix successfully. >>

```
[<<<<<< ChAMP.SVD END >>>>>>]
```

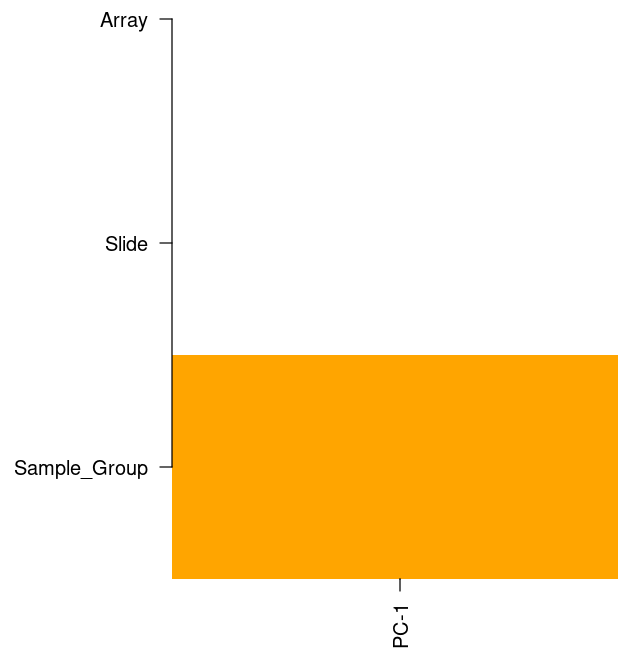
```
[=====]
```

[If the batch effect is not significant, you may want to process `champ.DMP()` or `champ.DMR()` or `champ.BlockFinder()` next, otherwise, you may want to run `champ.runCombat()` to eliminat batch effect, then rerun `champ.SVD()` to check corrected result.]

A matrix: 1 × 3 of type dbl

Sample_Group	Slide	Array
0.003609347	0.9074465	0.4049149

Singular Value Decomposition Analysis (SVD



Bestimmung und Visualisierung von differentiell methylierten Sonden (DMPs)



Nach erfolgreichem Laden der Daten, anschließendem Normalisieren und dem entfernen von Batch-Effekten kann jetzt final die Methylierung zwischen Gruppen von Interesse verglichen werden.

Dafür nutzen wir `champ.DMP`, was die Methylierung jeder Sonde zwischen zwei Gruppen (hier Samplegroups) vergleicht und testet, ob der Unterschied der Methylierung der Sonde zwischen den gruppen statistisch signifikant ist. Dabei kann jeweils nur ein Vergleich

zwischen zwei Gruppen stattfinden. Gibt es mehr als zwei Samplegroups, werden alle paarweisen Vergleiche berechnet.

Die Visualisierung erfolgt hier nicht über Plots, die im Notebook sichtbar sind, sondern über eine interaktive HTML Darstellung, die **über die angezeigte URL** erreichbar ist.

Die daraufhin angezeigte Seite beinhaltet verschiedene Elemente:

- **Eingabemaske (linker Rand)**: Hier müssen Nutzer einen **P-Cutoff** und einen **FoldChange** auswählen und auf **Submit** klicken, um DMPs anzeigen zu lassen. Über **Gene Symbol** kann man ein Gen auswählen, was daraufhin im Reiter **Gene** angezeigt wird. Über **CpG ID Symbol** kann man eine Sonde auswählen, die daraufhin im Reiter **CpG** einzeln angezeigt wird.
- **DMPtable**: Informationen zu den ausgewählten DMPs
- **Heatmap**: DMPs, die die SampleGroups am stärksten separieren
- **Feature&Cgi**: Positionen von DMPs in der DNA bzw. in CpG-Inseln
- **Gene**: Methylierung der DMPs eines Gens und Infos zu dem gewählten Gen
- **CpG**: Gene mit den meisten DMPs und einzelne CpGs

```
In [13]: DMPs <- champ.DMP(beta = myCombat, pheno=myLoad$pd$Sample_Group, adjPVal = 0.
DMP.GUI(DMP=DMPs[[1]], beta=myCombat, pheno=myLoad$pd$Sample_Group)
```

```
[=====]
```

```
[<<<<< ChAMP.DMP START >>>>>]
```

```
-----
```

!!! Important !!! New Modification has been made on champ.DMP():

(1): In this version champ.DMP() if your pheno parameter contains more than two groups of phenotypes, champ.DMP() would do pairwise differential methylated analysis between each pair of them. But you can also specify compare.group to only do comparison between any two of them.

(2): champ.DMP() now support numeric as pheno, and will do linear regression on them. So covariates like age could be inputted in this function. You need to make sure your inputted "pheno" parameter is "numeric" type.

```
-----
```

```
[ Section 1: Check Input Pheno Start ]
```

You pheno is character type.

Your pheno information contains following groups. >>

<Cellline>:12 samples.

<Coriell>:4 samples.

[The power of statistics analysis on groups contain very few samples may not strong.]

pheno contains only 2 phenotypes

compare.group parameter is NULL, two pheno types will be added into Compare List.

Cellline_to_Coriell compare group : Cellline, Coriell

```
[ Section 1: Check Input Pheno Done ]
```

```
[ Section 2: Find Differential Methylated CpGs Start ]
```

```
-----
```

Start to Compare : Cellline, Coriell

Contrast Matrix

Levels	Contrasts
	pCoriell-pCellline
pCellline	-1
pCoriell	1

You have found 685430 significant MVPs with a BH adjusted P-value below 0.05.

Calculate DMP for Cellline and Coriell done.

[Section 2: Find Numeric Vector Related CpGs Done]

[Section 3: Match Annotation Start]

[Section 3: Match Annotation Done]

[<<<<< ChAMP.DMP END >>>>>]

[=====]

[You may want to process DMP.GUI() or champ.GSEA() next.]

Lade nötiges Paket: shiny

Attache Paket: 'shiny'

Die folgenden Objekte sind maskiert von 'package:DT':

dataTableOutput, renderDataTable

Listening on http://127.0.0.1:3016

P value Cutoff:

abs(logFC) Cutoff:

Gene Symbol:

CpG ID:

Showing 1 to 20 of 685,430 entries

P value Cutoff:

abs(logFC) Cutoff:

Gene Symbol:

CpG ID:

NA12873 Jurkat NA12873 Jurkat NA12873 Jurkat NA12873 Jurkat NA12873 HeLa NA12873 HeLa NA12873 HeLa

P value Cutoff:

0.05

abs(log₂FC) Cutoff:

0.1

Submit

Gene Symbol:

CLDN10

Submit

CpG ID:

cg25032595_BC11

Submit



DMP Overview

P value Cutoff:

abs(logFC) Cutoff:

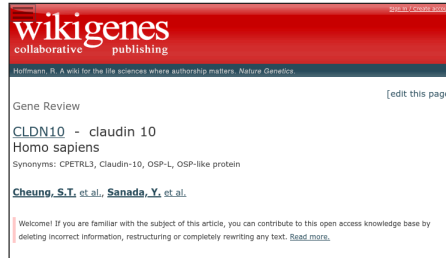
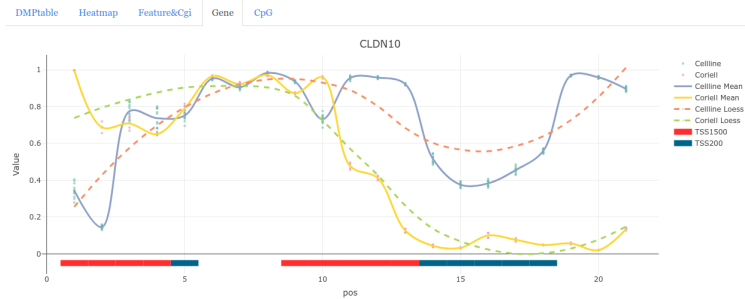
Submit

Gene Symbol:

Submit

CpG ID:

Submit



DMP Overview

P value Cutoff:

abs(logFC) Cutoff:

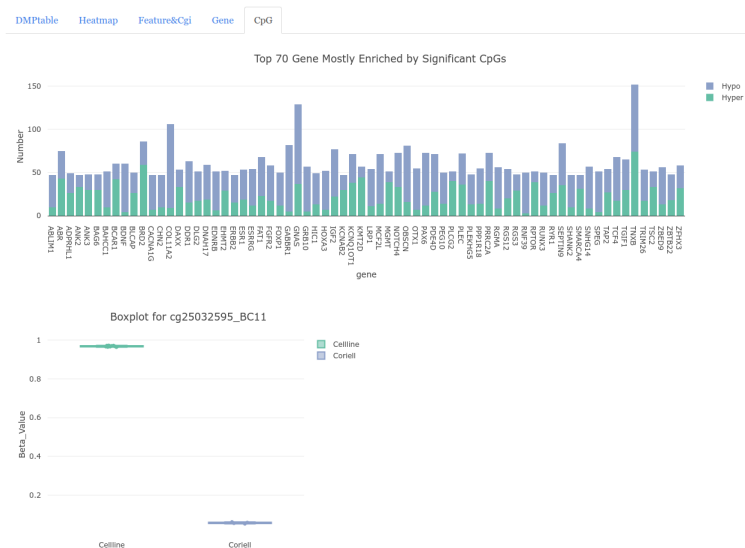
Submit

Gene Symbol:

Submit

CpG ID:

Submit



Bestimmung und Visualisierung von differentiell methylierten Regionen (DMRs)

Nach der Evaluation von DMPs können diese zu DMRs zusammengefasst werden. Eine DMR ist eine Region, in der sich besonders viele DMPs befinden.

Die Visualisierung ist ähnlich der von DMPs in verschiedene Elemente aufgeteilt:

- **Eingabemaske (linker Rand):** Hier müssen Nutzer einen **P-Cutoff** und eine Mindestanzahl an enthaltenen DMPs angeben werden, um das Set der gefundenen DMRs vorzufiltern. Über den **DMR-Index** kann eine einzelne DMR anhand ihrer ID in **DMRtable** ausgewählt werden. Die gewählte DMR wird dann im **DMRPlot** visualisiert
- **DMRtable:** Zeigt die ausgewählten DMRs und viele Zusatzinformationen an

- **CpGtable**: Durchsuchbare DMP Tabelle in der die Zuordnung der DMPs zu den DMRs einsehbar ist. Sehr praktisch, um DMRs zu suchen, die ein bestimmtes Gen enthalten, da **gene** nur in CpGtable enthalten ist
- **DMRPlot**: Anzeige einer bestimmten DMR nach Selektion über die **Eingabemaske**
- **Summary**: Zeigt die Gene an, für die die meisten CpGs in einer oder mehrerer DMRs liegen, sowie der DMP-Status dieser CpGs. Zusätzlich sind die größten Methylierungsunterschiede zwischen CpGs in DMRs der **Sample_Groups** dargestellt

```
In [14]: DMRs <- champ.DMR(beta=myCombat,pheno=myLoad$pd$Sample_Group,method="Bumphur")
          DMR.GUI(DMR=DMRs, arraytype="EPICv2")
```

```
[=====]
```

```
[<<<<< ChAMP.DMR START >>>>>]
```

```
-----
```

!!! important !!! We just upgrade champ.DMR() function, since now champ.DMP() could works on multiple phenotypes, but ProbeLasso can only works on one DMP result, so if your pheno parameter contains more than 2 phenotypes, and you want to use ProbeLasso function, you MUST specify compare.group=c("A","B"). Bumhunter and DMRcate should not be influenced.

```
[ Section 1: Check Input Pheno Start ]
```

You pheno is character type.

Your pheno information contains following groups. >>

<Cellline>:12 samples.

<Coriell>:4 samples.

```
[ Section 1: Check Input Pheno Done ]
```

```
[ Section 2: Run DMR Algorithm Start ]
```

<< Find DMR with Bumhunter Method >>

3 cores will be used to do parallel Bumhunter computing.

According to your data set, champ.DMR() detected 11309 clusters contains MORE THAN 7 probes within 300 maxGap. These clusters will be used to find DMR.

```
[bumhunterEngine] Parallelizing using 3 workers/cores (backend: doParallelM  
C, version: 1.0.17).
```

```
[bumhunterEngine] Computing coefficients.
```

```
[bumhunterEngine] Smoothing coefficients.
```

```
Lade nötiges Paket: rngtools
```

```
[bumhunterEngine] Performing 250 bootstraps.
```

```
[bumhunterEngine] Computing marginal bootstrap p-values.
```

```
[bumhunterEngine] Smoothing bootstrap coefficients.
```

```
[bumhunterEngine] cutoff: 0.184
```

[bumphunterEngine] Finding regions.

[bumphunterEngine] Found 16630 bumps.

[bumphunterEngine] Computing regions for each bootstrap.

[bumphunterEngine] Estimating p-values and FWER.

<< Calculate DMR success. >>

Bumphunter detected 13490 DMRs with P value ≤ 0.05 .

[Section 2: Run DMR Algorithm Done]

[<<<<< ChAMP.DMR END >>>>>]

[=====]

[You may want to process DMR.GUI() or champ.GSEA() next.]

!!! important !!! Since we just upgrated champ.DMP() function, which is now can support multiple phenotypes. Here in DMR.GUI() function, if you want to use "runDMP" parameter, and your pheno contains more than two groups of phenotypes, you MUST specify compare.group parameter as compare.group=c("A","B") to get DMP value between group A and group B.

[Section 1: Calculate DMP Start]

You pheno is character type.

Your pheno information contains following groups. >>

<Cellline>:12 samples.

<Coriell>:4 samples.

Your pheno contains EXACTLY two phenotypes, which is good, compare.group is not needed.

Calculating DMP

[=====]

[<<<<< ChAMP.DMP START >>>>>]

!!! Important !!! New Modification has been made on champ.DMP():

(1): In this version champ.DMP() if your pheno parameter contains more than two groups of phenotypes, champ.DMP() would do pairwise differential methylated analysis between each pair of them. But you can also specify compare.group to only do comparison between any two of them.

(2): champ.DMP() now support numeric as pheno, and will do linear regression on them. So covariates like age could be inputted in this function. You need to make sure your inputted "pheno" parameter is "numeric" type.

[Section 1: Check Input Pheno Start]

You pheno is character type.

Your pheno information contains following groups. >>

<Cellline>:12 samples.

<Coriell>:4 samples.

[The power of statistics analysis on groups contain very few samples may not strong.]

pheno contains only 2 phenotypes

compare.group parameter is NULL, two pheno types will be added into Compare List.

Cellline_to_Coriell compare group : Cellline, Coriell

[Section 1: Check Input Pheno Done]

[Section 2: Find Differential Methylated CpGs Start]

Start to Compare : Cellline, Coriell

Contrast Matrix

Levels	Contrasts
	pCoriell-pCellline
pCellline	-1
pCoriell	1

You have found 832552 significant MVPs with a BH adjusted P-value below 1.
Calculate DMP for Cellline and Coriell done.

[Section 2: Find Numeric Vector Related CpGs Done]

[Section 3: Match Annotation Start]

[Section 3: Match Annotation Done]

[<<<<<< ChAMP.DMP END >>>>>>]

[=====]

[You may want to process DMP.GUI() or champ.GSEA() next.]

[Section 1: Calculate DMP Done]

[Section 2: Mapping DMR to annotation Start]

Generating Annotation File

Generating Annotation File Success

[Section 2: Mapping DMR to annotation Done]

Listening on <http://127.0.0.1:3016>

BumphunterDMR Overview

P value Cutoff:
5 0.00 1

Min Number Probes:
100

Submit

DMR Index:
5 0.00 13.99

Submit

DMRTable		CpGTable	DMRPlot	Summary											
Show 20 entries		Search: <input type="text"/>													
ID	seqnames	start	end	width	strand	value	area	cluster	IndexStart	IndexEnd	L	cluster	p-value	fewer	p-value
DMR_1	DMR_1	chr8	31681093	31681951	1658	*	-7.713168211792694	262.247779920009316	495539	110882	110915	34	34	0	0
DMR_2	DMR_2	chr11	2139145	2141300	2055	*	-5.449484835916576	256.13214742880696	74390	20034	20080	47	47	0	0
DMR_3	DMR_3	chr7	9454949	9457643	2697	*	-3.676073433215575	255.3149349356937	441137	122236	122318	83	86	0	0
DMR_4	DMR_4	chr12	132887021	132888427	1406	*	-8.48630720782542	246.0997190269436	126507	35540	35568	29	29	0	0
DMR_5	DMR_5	chr4	153789192	15379754	1682	*	-6.49061208192007	235.4429234985145	397327	98809	98844	36	36	0	0
DMR_6	DMR_6	chr6	30100716	30102565	1849	*	-7.079906181025452	219.1981538715973	49454	10839	10869	31	31	0	0
DMR_7	DMR_7	chr10	132786279	132787415	1136	*	-8.593494180781601	212.5830431954	72599	19470	19494	25	25	0	0
DMR_8	DMR_8	chr6	39913335	39915426	2091	*	-4.79155822387271	212.466781379317	495236	110146	110190	45	45	0	0
DMR_9	DMR_9	chr6	30074360	30076482	2122	*	-5.431162943020699	211.8153547778073	494845	108900	108938	39	39	0	0
DMR_10	DMR_10	chr11	44399353	44311642	2289	*	-4.774322835530367	205.39144819365588	81487	22238	22280	43	43	0	0
DMR_11	DMR_11	chr6	33279575	33278711	1736	*	-4.337582041347348	201.8663559433253	495232	113434	113480	47	47	0	0
DMR_12	DMR_12	chr6	28634766	28633660	894	*	-8.123364439161685	203.0841814395546	494381	108130	108154	35	35	0	0
DMR_13	DMR_13	chr6	133245230	133241638	1408	*	-6.548324675590637	202.998014349197	418798	117372	117392	31	31	0	0
DMR_14	DMR_14	chr10	58859924	58833418	2494	*	-3.221794413108064	199.7510057367	399997	82314	82375	62	62	0	0
DMR_15	DMR_15	chr6	42961002	42961182	980	*	-7.698167614228865	192.451621232162	499328	114759	114763	35	35	0	0
DMR_16	DMR_16	chr12	114695396	114698593	2107	*	-3.975525198871847	186.8320343496768	120732	34473	34439	47	47	0	0
DMR_17	DMR_17	chr6	30006245	30007898	1653	*	-7.059593653218321	182.1543581967674	494782	108786	108811	26	26	0	0
DMR_18	DMR_18	chr7	27167377	27170655	2278	*	-6.3721551762866	178.4303449549005	433188	120220	120247	28	28	0	0
DMR_19	DMR_19	chr6	29552921	29554104	1083	*	-5.904592191896424	177.1968652568927	494616	108515	108544	30	30	0	0
DMR_20	DMR_20	chr18	77249673	77251413	1740	*	-5.876572121918907	170.4405915359395	326607	66661	66689	29	29	0	0

Showing 1 to 20 of 6,887 entries

Previous

12345—350Next

Showing 1 to 20 of 6,987 entries

Previous 1 2 3 4 5 ... 350 Next

BumphunterDMR Overview

P value Cutoff:
5 0.00 1

Min Number Probes:
100

Submit

DMR Index:
5 0.00 13.99

Submit

DMRTable	CpGTable	DMRPlot	Summary									
Show 20 entries		Search:										
ID	DMRIndex	CHR	MAPINFO	Strand	Type	gene	feature	egl	feat.egl	t	adj.P.Val	
cg09992780_BC21	cg09992780_BC21	DMR_1	chr6	31681957	R	II	LY6G3C	Exon	Shore	Exon-Shore	-1.755329648531018	0.2487915797879214
cg08623951_TC21	cg08623951_TC21	DMR_1	chr6	31681084	F	II	LY6G3C	Exon	Shore	Exon-Shore	-1.321854142744341	0.3772955531082244
cg20543183_BC21	cg20543183_BC21	DMR_1	chr6	31681039	R	II	LY6G3C	Exon	Shore	Exon-Shore	-0.3608728399618769	0.8281972123284235
cg00027925_BC21	cg00027925_BC21	DMR_1	chr6	31681052	F	II	LY6G3C	Exon	Shore	Exon-Shore	-1.393327335593935	0.3964274333351966
cg21036162_BC21	cg21036162_BC21	DMR_1	chr6	31681935	F	II	LY6G3C	TSR1300	Shore	TSR1300-Shore	-38.337814938766727	1.4944200538954359-14
cg03588345_TC21	cg03588345_TC21	DMR_1	chr6	31681171	R	II	LY6G3C	TSR200	Shore	TSR200-Shore	-2.490396510423287	0.03668439199798883
cg23642766_BC21	cg23642766_BC21	DMR_1	chr6	31681176	R	II	LY6G3C	TSR200	Shore	TSR200-Shore	-2.385571462147793	0.115321249944723
cg02615735_BC21	cg02615735_BC21	DMR_1	chr6	31681499	R	II	LY6G3C	Exon	Shore	Exon-Shore	-18.126620651493	5.10149777697142-10
cg07845496_TC21	cg07845496_TC21	DMR_1	chr6	31681023	R	II	LY6G3C	Exon	Shore	Exon-Shore	-1.4114145398854	0.33953779474357142
cg27444142_BC21	cg27444142_BC21	DMR_1	chr6	31681023	F	II	LY6G3C	Exon	Shore	Exon-Shore	1.628031280154808	0.205565622371741
cg09128167_BC21	cg09128167_BC21	DMR_1	chr6	31681593	R	II	LY6G3C	TSR1300	Inland	TSR1300-Inland	-2.02619539198665571	0.1798280338144363
cg0412885_BC21	cg0412885_BC21	DMR_1	chr6	31681702	R	II	LY6G3C	TSR1300	Shore	TSR1300-Shore	-9.104723236564356	0.000000947954493276335
cg05541966_TC21	cg05541966_TC21	DMR_1	chr6	31681737	R	II	LY6G3C	Exon	Shore	Exon-Shore	-1.97957414937872	0.19495246217750908
cg02683378_TC21	cg02683378_TC21	DMR_1	chr6	31681653	F	II	LY6G3C	TSR1300	Shore	TSR1300-Shore	-5.197979234389451	0.0011204817659991
cg06667222_TC21	cg06667222_TC21	DMR_1	chr6	31681217	F	II	LY6G3C	TSR200	Shore	TSR200-Shore	-2.7653792737787671	0.06768427851399174
cg11679268_BC21	cg11679268_BC21	DMR_1	chr6	31681864	R	II	LY6G3C	TSR1300	Shore	TSR1300-Shore	0.913919133772283	0.549874318987431
cg20717622_TC21	cg20717622_TC21	DMR_1	chr6	31681849	F	II	LY6G3C	Exon	Shore	Exon-Shore	-1.355812739081899	0.370706902542614

Showing 1 to 13 of 13 entries

BumphunterDMR Overview

P value Cutoff:
5 0.00 1

Min Number Probes:
100

Submit

DMR Index:
5 0.00 13.99

Submit

DMRTableCpGTableDMRPlotSummary

Show8▼entries

Search:

ID	DMRIndex	CHR	MAPINFO	Strand	Type	gene	feature	egl	feat.egl	
cg0467067_TC21	cg0467067_TC21	DMR_912	chr13	95532011	R	II	CLDN30	TSR1300	Shore	TSR1300-Shore
cg00022500_TC21	cg00022500_TC21	DMR_912	chr13	95532190	F	II	CLDN30	TSR1300	Shore	TSR1300-Shore
cg18470456_BC21	cg18470456_BC21	DMR_912	chr13	95532239	R	II	CLDN30	TSR1300	Shore	TSR1300-Shore
cg13431616_TC21	cg13431616_TC21	DMR_912	chr13	95532264	R	II	CLDN30	TSR1300	Shore	TSR1300-Shore
cg4426691_TC11	cg4426691_TC11	DMR_912	chr13	95532415	R	I	CLDN30	TSR1300	Shore	TSR1300-Shore
cg16231813_TC21	cg16231813_TC21	DMR_912	chr13	95532600	F	II	CLDN30	TSR200	Inland	TSR200-Inland
cg04281613_BC21	cg04281613_BC21	DMR_912	chr13	95532612	F	II	CLDN30	TSR200	Inland	TSR200-Inland
cg06757399_TC11	cg06757399_TC11	DMR_912	chr13	95532606	F	I	CLDN30	TSR200	Inland	TSR200-Inland

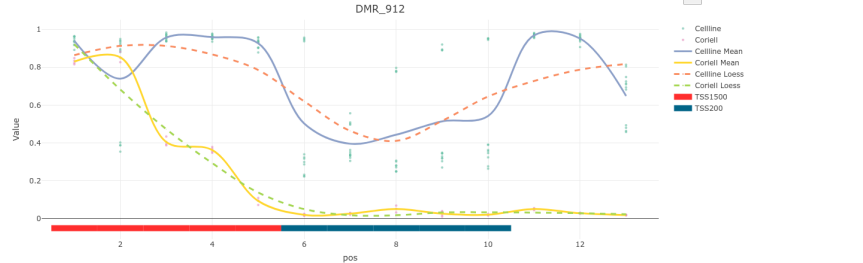
Showing 1 to 8 of 13 entries

Previous

12Next

Showing 1 to 8 of 13 entries

Previous 1 2 Next



BumphunterDMR Overview

DMRtable

CpGtable

DMRPlot

Summary

P value Cutoff:

0.05

1

Min Number Probes:

10

100

Submit

DMR Index:

0.2

13.269

Submit

