

2018.06.12版

スライド32まではさらっといきます(スライド4-6, 13, 18, 28のみしゃべる予定)。6/19出席予定の持込PCの方は、Rパッケージrecountをインストールしておいてください。

農学生命情報科学特論I 第1回

¹大学院農学生命科学研究科
アグリバイオインフォマティクス教育研究プログラム

²微生物科学イノベーション連携研究機構

門田幸二(かどた こうじ)

kadota@iu.a.u-tokyo.ac.jp

<http://www.iu.a.u-tokyo.ac.jp/~kadota/>

講義予定

- 第1回(2018年06月12日)
 - カウント情報取得の続き
 - データの正規化(RPK, RPM, RPKM/FPKM)
- 第2回(2018年06月19日)
 - サンプル間クラスタリング、結果の客観的な評価(Silhouetteスコア)
 - クラスタリング結果の客観的な評価
- 第3回(2018年06月26日)
 - 発現変動解析(多重比較問題とFDR)、各種プロット(M-A plot)
 - 発現変動解析(デザイン行列や3群間比較)
- 第4回(2018年07月03日)
 - 機能解析(Gene Ontology解析やパスウェイ解析)

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イントロ、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

おさらい

①のsingle-endでアノテーション有の、②例題10の実行結果として、365遺伝子のカウントデータしか得られなかった。機能ゲノム学第4回のスライド82-99

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページは、フリーソフトRとWindows2015 (2015/04/03)

- マップ後 | 出力ファイルの読み込み | [htSeqTools\(Planet 2012\)](#) (last modified 2013/06/19)
- マップ後 | [カウント情報取得](#) | [|について](#) (last modified 2018/05/30) **NEW**
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/29) **①**
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [HTSeq\(Anders 2015\)](#) (last modified 2018/05/30) **N**
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション無 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/26)
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2016/02/12)

What's new?

- 「マップ後」
- 「イントロ」
- 「H29年度N

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR (Gaidatzis_2015) **NEW**

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カウントデータ取得までの一連の流れを示します。アノテーション情報は、GenomicFeaturesパッケージ中の関数を利用してTxDbオブジェクトをネットワーク経由で取得するのを基本としつつ、TxDbパッケージを読み込むやり方も示しています。マッピングのやり方やアノテーションの詳細については、マッピング(single-end | ゲノム | アノテーション有) | QuasR(Gaidatzis_2015)などを参考にしてください。

② 10.mapping single genome7.txt中のFASTA形式ファイルを乳酸菌ゲノムにマッピングする場合:

マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl (Zerbino et al., Nucleic Acids Res., 2018)から提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。マッピング結果に対して、GFF3形式のアノテーションファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3)を読み込んでカウント情報を取得しています。

```
in_f1 <- "mapping_single_genome7.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome
in_f3 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff
out_f <- "hoge10.txt" #出力ファイル名を指定してout_fに格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定: "gene", "exon", "pro

#必要なパッケージをロード
library(QuasR) #パッケージの読み込み
library(GenomicFeatures) #パッケージの読み込み
```

おさらい

当該GFF3ファイルの中身。①Name=の右側の文字が `genename`。②この遺伝子領域には `Name=genename` がないこともわかる。これらが原因で2000行超にはならず365行となってしまったのかも…と考えた。

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939v1
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
Chromosome ena gene 360 1676 . + . ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;d
Chromosome ena transcript 360 1676 . + . ID=transcript:BAP84581;Parent=gene:LOOC260_100010;Name=d
Chromosome ena exon 360 1676 . + . Parent=transcript:BAP84581;Name=BAP84581-1;constitutive=1;e
Chromosome ena CDS 360 1676 . + 0 ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_id=BAP8
###
Chromosome ena gene 1852 2991 . + . ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding;d
Chromosome ena transcript 1852 2991 . + . ID=transcript:BAP84582;Parent=gene:LOOC260_100020;Name=d
Chromosome ena exon 1852 2991 . + . Parent=transcript:BAP84582;Name=BAP84582-1;constitutive=1;e
Chromosome ena CDS 1852 2991 . + 0 ID=CDS:BAP84582;Parent=transcript:BAP84582;protein_id=BAP8
###
Chromosome ena gene 3233 3457 . - ID=gene:LOOC260_100030;biotype=protein_coding;description=S4
Chromosome ena transcript 3233 3457 . + . ID=transcript:BAP84583;Parent=gene:LOOC260_100030;biotype=
Chromosome ena exon 3233 3457 . + . Parent=transcript:BAP84583;Name=BAP84583-1;constitutive=1;e
Chromosome ena CDS 3233 3457 . + 0 ID=CDS:BAP84583;Parent=transcript:BAP84583;protein_id=BAP8
###
Chromosome ena gene 3467 4588 . + . ID=gene:LOOC260_100040;Name=recF;biotype=protein_coding;de
```



おさらい

機能ゲノム学第4回のスライド130-141をまとめたもの。①GFF3ファイルからID=geneを含む行をuge.txtに保存。②uge.txtの行数は2262。③uge.txtにおいて、Name=を含む行数は457。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [10:19午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [10:19午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt ① [10:20午前]
iu@bielinux[mapping_kiso3] wc uge.txt [10:20午前]
2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [10:20午前]
457
iu@bielinux[mapping_kiso3] [10:21午前]
```


Tips: パイプ (|)

①と②を(uge.txtのような中間ファイルを作成せずに)行うやり方を説明します。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [10:19午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [10:19午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt ①
iu@bielinux[mapping_kiso3] wc uge.txt [10:20午前]
2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [10:20午前]
457
iu@bielinux[mapping_kiso3] [10:21午前]
```

Tips: パイプ (|)

①と②を(uge.txtのような中間ファイルを作成せずに)行うやり方を説明します。③パイプ(縦棒の|、のこと)を使うことで1行で書けます。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [10:19午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [10:19午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt ①
iu@bielinux[mapping_kiso3] wc uge.txt [10:20午前]
2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [10:20午前]
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | wc ③
2262 24154 398272
iu@bielinux[mapping_kiso3] [11:11午前]
```


Tips: パイプ (|)

④が①の部分に相当し、⑤が②の部分に相当します。中間ファイルのuge.txtを作成していないことがわかります。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [10:19午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [10:19午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt ①
iu@bielinux[mapping_kiso3] wc uge.txt [10:20午前]
2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [10:20午前]
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | wc ④ ⑤
2262 24154 398272
iu@bielinux[mapping_kiso3] [11:11午前]
```

もう1つの例。①と②をパイプで連結したのが③の
コマンド。同じ結果になっていることがわかります。

Tips: パイプ (|)

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [11:32午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [11:32午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
① iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt
iu@bielinux[mapping_kiso3] wc uge.txt [11:32午前]
2262 24154 398272 uge.txt
② iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [11:32午前]
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | wc
2262 24154 398272
③ iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep -c "Name="
457
iu@bielinux[mapping_kiso3] [11:34午前]
```

行数を減らして眺める

①はID=geneとName=を含む行を②ukyo.txtに保存するコマンド

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [11:32午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [11:32午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt
iu@bielinux[mapping_kiso3] wc uge.txt [11:32午前]
2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [11:32午前]
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | wc
2262 24154 398272
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep -c "Name="
457
① iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
> ukyo.txt
iu@bielinux[mapping_kiso3] [11:44午前]
②
```


行数を減らして眺める

- ①ukyo.txtをエクセルで眺めているところ。
- ②tRNA_geneを発見。
- ③対応するName=の右側の文字列も他のものとは趣が異なる。

	A	B	C	D	E	F	G	H	I
1	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;des
2	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding;des
3	Chromosome	ena	gene	3467	4588	.	+	.	ID=gene:LOOC260_100040;Name=recF;biotype=protein_coding;desc
4	Chromosome	ena	gene	4588	6531	.	+	.	ID=gene:LOOC260_100050;Name=gyrB;biotype=protein_coding;desc
5	Chromosome	ena	gene	6559	9120	.	+	.	ID=gene:LOOC260_100060;Name=gyrA;biotype=protein_coding;desc
6	Chromosome	ena	gene	10869	11165	.	+	.	ID=gene:LOOC260_100080;Name=rpsF;biotype=protein_coding;desc
7	Chromosome	ena	gene	11758	11994	.	+	.	ID=gene:LOOC260_100100;Name=rpsR;biotype=protein_coding;desc
8	Chromosome	ena	tRNA_gene	23781	23853	.	-	.	ID=gene:LOOC260_100220;Name=LOOC260_100220;biotype=tRNA;
9	Chromosome	ena	gene	31109	34192	.	-	.	ID=gene:LOOC260_100290;Name=carB;biotype=prot coding;desc
10	Chromosome	ena	gene	34185	35267	.	-	.	ID=gene:LOOC260_100300;Name=carA;biotype=protein_coding;desc
11	Chromosome	ena	gene	35510	36538	.	+	.	ID=gene:LOOC260_100310;Name=argC;biotype=protein_coding;desc
12	Chromosome	ena	gene	36551	37756	.	+	.	ID=gene:LOOC260_100320;Name=argJ;biotype=protein_coding;desc
13	Chromosome	ena	gene	37768	38511	.	+	.	ID=gene:LOOC260_100330;Name=argB;biotype=protein_coding;desc
14	Chromosome	ena	gene	38534	39670	.	+	.	ID=gene:LOOC260_100340;Name=argD;biotype=protein_coding;desc
15	Chromosome	ena	gene	39675	40694	.	+	.	ID=gene:LOOC260_100350;Name=argF;biotype=protein_coding;desc
16	Chromosome	ena	gene	54793	56562	.	-	.	ID=gene:LOOC260_100510;Name=horA;biotype=protein_coding;desc
17	Chromosome	ena	gene	56764	57642	.	+	.	ID=gene:LOOC260_100520;Name=pepIP;biotype=protein_coding;de
18	Chromosome	ena	tRNA_gene	140244	140315	.	+	.	ID=gene:LOOC260_101280;Name=LOOC260_101280;biotype=tRNA;

行数を減らして眺める

少し下のほうも探索。①rRNA_geneというのもありそう。おそらくQuasRは、tRNA_geneとrRNA_geneの情報を出力しないようにしているのだろう。というようなことを経験値として蓄積していく。

	A	B	C	D	E	F	G	H	I
16	Chromosome	ena	gene	54793	56562	.	-	.	ID=gene:LOOC260_100510;Name=horA;biotype=protein_coding;desc
17	Chromosome	ena	gene	56764	57642	.	+	.	ID=gene:LOOC260_100520;Name=pepIP;biotype=protein_coding;de
18	Chromosome	ena	tRNA_gene	140244	140315	.	+	.	ID=gene:LOOC260_101280;Name=LOOC260_101280;biotype=tRNA;
19	Chromosome	ena	gene	158199	158909	.	-	.	ID=gene:LOOC260_101520;Name=ubiE;biotype=protein_coding;desc
20	Chromosome	ena	gene	179616	180083	.	-	.	ID=gene:LOOC260_101730;Name=greA;biotype=protein_coding;desc
21	Chromosome	ena	gene	194973	196004	.	+	.	ID=gene:LOOC260_101870;Name=lplA;biotype=protein_coding;desc
22	Chromosome	ena	gene	199757	201103	.	+	.	ID=gene:LOOC260_101910;Name=trkH;biotype=protein_coding;desc
23	Chromosome	ena	gene	201126	201788	.	+	.	ID=gene:LOOC260_101920;Name=trkA;biotype=protein_coding;desc
24	Chromosome	ena	gene	227737	228603	.	+	.	ID=gene:LOOC260_102100;Name=parB;biotype=protein_coding;desc
25	Chromosome	ena	gene	228621	229388	.	+	.	ID=gene:LOOC260_102110;Name=parA;biotype=protein_coding;desc
26	Chromosome	ena	gene	229378	230259	.	+	.	ID=gene:LOOC260_102120;Name=parB;biotype=protein_coding;desc
27	Chromosome	ena	gene	247127	248362	.	+	.	ID=gene:LOOC260_102330;Name=gshA;biotype=protein_coding;desc
28	Chromosome	ena	gene	310153	310422	.	+	.	ID=gene:LOOC260_102950;Name=rpsN;biotype=protein_coding;desc
29	Chromosome	ena	rRNA_gene	310417	351989	.	+	.	ID=gene:LOOC260_103350;Name=LOOC260_103350;biotype=rRNA;
30	Chromosome	ena	rRNA_gene	310215	355138	.	+	.	ID=gene:LOOC260_103360;Name=LOOC260_103360;biotype=rRNA;
31	Chromosome	ena	rRNA_gene	315236	355351	.	+	.	ID=gene:LOOC260_103380;Name=LOOC260_103380;biotype=rRNA;
32	Chromosome	ena	gene	358113	359453	.	+	.	ID=gene:LOOC260_103410;Name=gshR;biotype=protein_coding;desc
33	Chromosome	ena	gene	393862	394773	.	+	.	ID=gene:LOOC260_103710;Name=cvsK;biotype=protein_coding;desc

①ukyo.txtの中から、grep -vでtRNA_geneと
rRNA_gene以外の行のみukyo2.txtに保存。

grep -v

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [11:32午前]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [11:32午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt
iu@bielinux[mapping_kiso3] wc uge.txt [11:32午前]
 2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [11:32午前]
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | wc
 2262 24154 398272
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep -c "Nam
e="
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
> ukyo.txt
iu@bielinux[mapping_kiso3] grep -v "tRNA_gene" ukyo.txt | grep
-v "rRNA_gene" > ukyo2.txt
iu@bielinux[mapping_kiso3] [ 1:07午後]
```



①wcでukyo2.txtの行数を確認。389行ですね。大分365行に近づいてきました。

457行から389行に

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] ls [11:32午前]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 > uge.txt
iu@bielinux[mapping_kiso3] wc uge.txt [11:32午前]
2262 24154 398272 uge.txt
iu@bielinux[mapping_kiso3] grep -c "Name=" uge.txt [11:32午前]
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | wc
2262 24154 398272
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep -c "Name="
457
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name=" > ukyo.txt
iu@bielinux[mapping_kiso3] grep -v "tRNA_gene" ukyo.txt | grep -v "rRNA_gene" > ukyo2.txt
① iu@bielinux[mapping_kiso3] wc ukyo2.txt [ 1:07午後]
389 4409 73753 ukyo2.txt
iu@bielinux[mapping_kiso3] [ 1:11午後]
```


ukyo2.txt

①wcでukyo2.txtの行数を確認。389行ですね。大分365行に近づいてきました。このあとは、②のあたりに着目しながら、何か変なところがないかを調べる。経験上、機能ゲノム学第4回の最後のスライド141でも言及しているように、同じ遺伝子名のものがあるのではという可能性を疑いながらざっと眺めると…

	A	B	C	D	E	F	G	H	I
1	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;des
2	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding;des
3	Chromosome	ena	gene	3467	4588	.	+	.	ID=gene:LOOC260_100040;Name=recF;biotype=protein_coding;des
4	Chromosome	ena	gene	4588	6531	.	+	.	ID=gene:LOOC260_100050;Name=gyrB;biotype=protein_coding;des
5	Chromosome	ena	gene	6559	9120	.	+	.	ID=gene:LOOC260_100060;Name=gyrA;biotype=protein_coding;des
6	Chromosome	ena	gene	10869	11165	.	+	.	ID=gene:LOOC260_100080;Name=rpsF;biotype=protein_coding;des
7	Chromosome	ena	gene	11758	11994	.	+	.	ID=gene:LOOC260_100100;Name=rpsR;biotype=protein_coding;des
8	Chromosome	ena	gene	31109	34192	.	-	.	ID=gene:LOOC260_100290;Name=carB;biotype=protein_coding;des
9	Chromosome	ena	gene	34185	35267	.	-	.	ID=gene:LOOC260_100300;Name=carA;biotype=protein_coding;des
10	Chromosome	ena	gene	35510	36538	.	+	.	ID=gene:LOOC260_100310;Name=argC;biotype=protein_coding;des
11	Chromosome	ena	gene	36551	37756	.	+	.	ID=gene:LOOC260_100320;Name=argJ;biotype=protein_coding;des
12	Chromosome	ena	gene	37768	38511	.	+	.	ID=gene:LOOC260_100330;Name=argB;biotype=protein_coding;des
13	Chromosome	ena	gene	38534	39670	.	+	.	ID=gene:LOOC260_100340;Name=argD;biotype=protein_coding;des
14	Chromosome	ena	gene	39675	40694	.	+	.	ID=gene:LOOC260_100350;Name=argF;biotype=protein_coding;des
15	Chromosome	ena	gene	54793	56562	.	-	.	ID=gene:LOOC260_100510;Name=horA;biotype=protein_coding;des
16	Chromosome	ena	gene	56764	57642	.	+	.	ID=gene:LOOC260_100520;Name=pepIP;biotype=protein_coding;d
17	Chromosome	ena	gene	158199	158909	.	-	.	ID=gene:LOOC260_101520;Name=ubiE;biotype=protein_coding;des
18	Chromosome	ena	gene	179616	180083	.	-	.	ID=gene:LOOC260_101730;Name=greA;biotype=protein_coding;des

ukyo2.txt

Excel spreadsheet showing genomic data from 'ukyo2.txt'. The spreadsheet has columns A through I. The data is organized into rows representing different genes on a chromosome.

	A	B	C	D	E	F	G	H	I
13	Chromosome	ena	gene	38534	39670	.	+	.	ID=gene:LOOC260_100340;Name=argD;biotype=protein_coding;de
14	Chromosome	ena	gene	39675	40694	.	+	.	ID=gene:LOOC260_100350;Name=argF;biotype=protein_coding;des
15	Chromosome	ena	gene	54793	56562	.	-	.	ID=gene:LOOC260_100510;Name=horA;biotype=protein_coding;des
16	Chromosome	ena	gene	56764	57642	.	+	.	ID=gene:LOOC260_100520;Name=pepIP;biotype=protein_coding;d
17	Chromosome	ena	gene	158199	158909	.	-	.	ID=gene:LOOC260_101520;Name=ubiE;biotype=protein_coding;des
18	Chromosome	ena	gene	179616	180083	.	-	.	ID=gene:LOOC260_101730;Name=greA;biotype=protein_coding;des
19	Chromosome	ena	gene	194973	196004	.	+	.	ID=gene:LOOC260_101870;Name=lpIA;biotype=protein_coding;des
20	Chromosome	ena	gene	199757	201103	.	+	.	ID=gene:LOOC260_101910;Name=trkH;biotype=protein_coding;des
21	Chromosome	ena	gene	201126	201788	.	+	.	ID=gene:LOOC260_101920;Name=trkA;biotype=protein_coding;des
22	Chromosome	ena	gene	227737	228603	.	+	.	ID=gene:LOOC260_102100;Name= <u>parB</u> ;biotype=protein_coding;de
23	Chromosome	ena	gene	228621	229388	.	+	.	ID=gene:LOOC260_102110;Name=parA;biotype=protein_coding;de
24	Chromosome	ena	gene	229378	230259	.	+	.	ID=gene:LOOC260_102120;Name= <u>parB</u> ;biotype=protein_coding;de
25	Chromosome	ena	gene	247127	248362	.	+	.	ID=gene:LOOC260_102330;Name=gshA;biotype=protein_coding;de
26	Chromosome	ena	gene	310153	310422	.	+	.	ID=gene:LOOC260_102950;Name=rpsN;biotype=protein_coding;de
27	Chromosome	ena	gene	358113	359453	.	+	.	ID=gene:LOOC260_103410;Name=gshR;biotype=protein_coding;de
28	Chromosome	ena	gene	393862	394773	.	+	.	ID=gene:LOOC260_103710;Name=cysK;biotype=protein_coding;de
29	Chromosome	ena	gene	394790	395929	.	+	.	ID=gene:LOOC260_103720;Name=metC;biotype=protein_coding;de
30	Chromosome	ena	gene	399136	399984	.	+	.	ID=gene:LOOC260_103770;Name=nurR;biotype=protein_coding;de

Red circles with the number '1' highlight the 'parB' gene names in rows 22 and 24, indicating a duplicate.

ukyo2.txt

①ここにも!。というわけで、残りは②のところの情報のみ抽出して、ユニークな遺伝子数が365になることを確認する作業に移行。無事365行になれば、QuasRを用いた遺伝子レベルのカウントデータは、そういう内部的な処理をしているという理解につながる。例えば、tRNAやrRNAのカウントデータは取得していない、など。

	A	B	C	D	E	F	G	H	I
145	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110170;Name=xseB;biotype=protein_coding;de
146	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110200;Name=argR;biotype=protein_coding;de
147	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110280;Name=rpoZ;biotype=protein_coding;de
148	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110360;Name=rpe;biotype=protein_coding;desc
149	Chromosome	ena	gene	1E+06	1E+06	.	-	.	ID=gene:LOOC260_110380;Name=rpmB;biotype=protein_coding;de
150	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110420;Name=plsX;biotype=protein_coding;des
151	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110440;Name=rnc;biotype=protein_coding;desc
152	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110490;Name=xpk;biotype=protein_coding;desc
153	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110510;Name=pntA;biotype=protein_coding;de
154	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110520;Name=pntA;biotype=protein_coding;de
155	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110530;Name=pntB;biotype=protein_coding;de
156	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110540;Name=rpsP;biotype=protein_coding;de
157	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110560;Name=rpmM;biotype=protein_coding;de
158	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110570;Name=trmD;biotype=protein_coding;de
159	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110580;Name=rplS;biotype=protein_coding;des
160	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110610;Name=lysA;biotype=protein_coding;des
161	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110620;Name=glpK;biotype=protein_coding;des
162	Chromosome	ena	gene	1E+06	1E+06	.	-	.	ID=gene:LOOC260_110680;Name=mvaD;biotype=protein_coding;de

cutコマンド

まずは、cutコマンドで特定の列のみ取り出す。①は9列目なので…

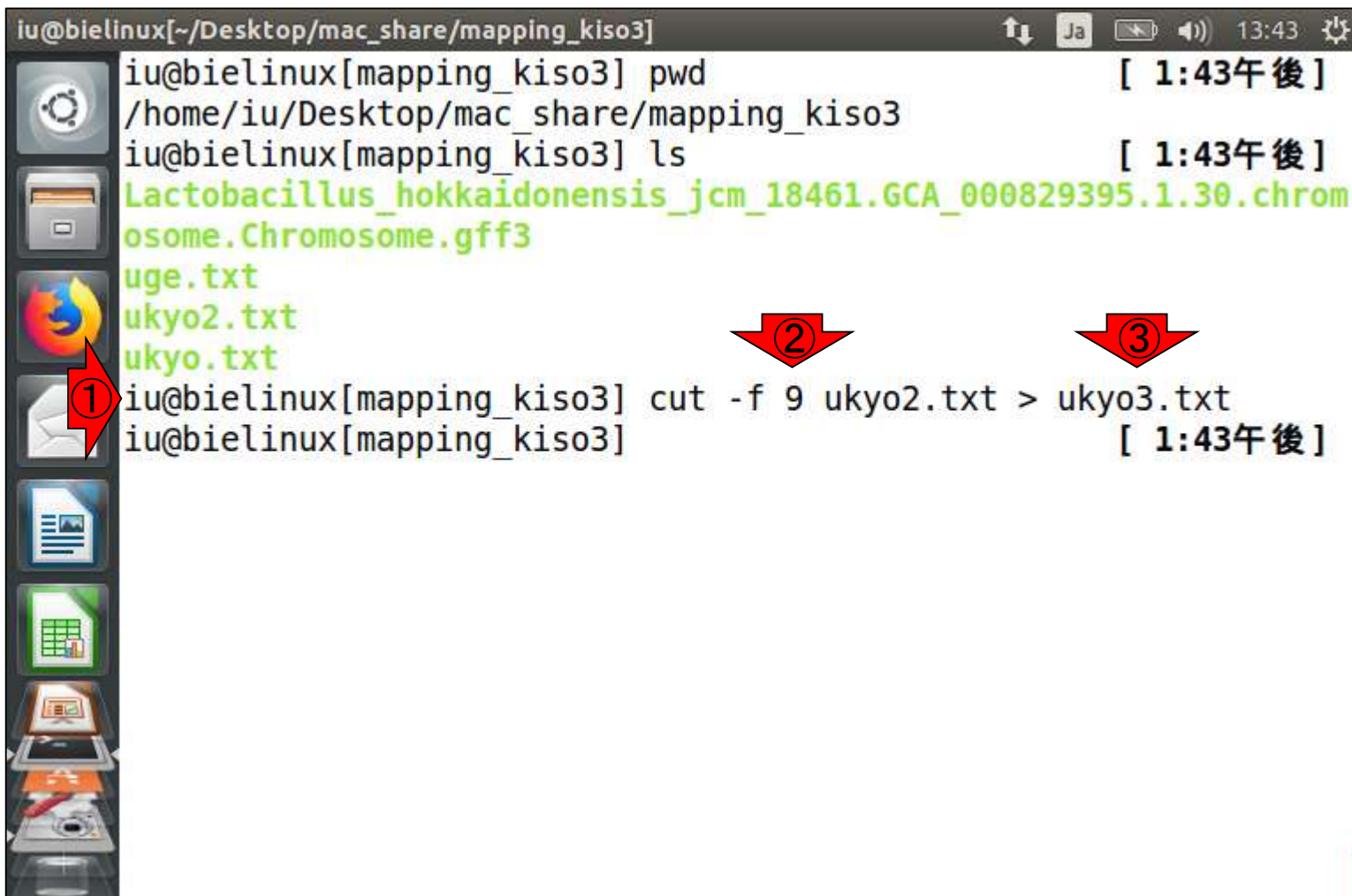
The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I
145	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110170;Name=xseB;biotype=protein_coding;de
146	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110200;Name=argR;biotype=protein_coding;de
147	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110280;Name=rpoZ;biotype=protein_coding;de
148	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110360;Name=rpe;biotype=protein_coding;desc
149	Chromosome	ena	gene	1E+06	1E+06	.	-	.	ID=gene:LOOC260_110380;Name=rpmB;biotype=protein_coding;de
150	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110420;Name=plsX;biotype=protein_coding;des
151	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110440;Name=rnc;biotype=protein_coding;desc
152	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110490;Name=xpk;biotype=protein_coding;desc
153	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110510;Name=pntA;biotype=protein_coding;de
154	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110520;Name=pntA;biotype=protein_coding;de
155	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110530;Name=pntB;biotype=protein_coding;de
156	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110540;Name=rpsP;biotype=protein_coding;de
157	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110560;Name=rpmM;biotype=protein_coding;de
158	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110570;Name=trmD;biotype=protein_coding;de
159	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110580;Name=rplS;biotype=protein_coding;des
160	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110610;Name=lysA;biotype=protein_coding;des
161	Chromosome	ena	gene	1E+06	1E+06	.	+	.	ID=gene:LOOC260_110620;Name=glpK;biotype=protein_coding;des
162	Chromosome	ena	gene	1E+06	1E+06	.	-	.	ID=gene:LOOC260_110680;Name=mvaD;biotype=protein_coding;de

cutコマンド

①こんな感じで、②9列目の情報を取り出した結果を、③ukyo3.txtに保存。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [ 1:43午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 1:43午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo.txt
iu@bielinux[mapping_kiso3] cut -f 9 ukyo2.txt > ukyo3.txt [ 1:43午後 ]
iu@bielinux[mapping_kiso3]
```



ukyo3.txt

ずらずらと長いので、区切り文字を探す。①セミコロン;にすればよさそう。そうすると、分割後の②1番目のフィールド、③2番目のフィールド、④3番目のフィールドみたいになるので、2番目のフィールドを抽出する。

The screenshot shows a spreadsheet with a single column of text. The text consists of gene identifiers and their descriptions, separated by semicolons. Red arrows point to specific parts of the first row:

- Arrow ① points to the semicolon after the gene ID.
- Arrow ② points to the gene name.
- Arrow ③ points to the biotype.
- Arrow ④ points to the description.

	A	B	C
1	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein Dn		
2	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding;description=DNA polymerase III subunit beta;gene_id=LO		
3	ID=gene:LOOC260_100040;Name=recF;biotype=protein_coding;description=DNA replication and repair protein RecF;gene,		
4	ID=gene:LOOC260_100050;Name=gyrB;biotype=protein_coding;description=DNA gyrase subunit B;gene_id=LOOC260_100		
5	ID=gene:LOOC260_100060;Name=gyrA;biotype=protein_coding;description=DNA gyrase subunit A;gene_id=LOOC260_100		
6	ID=gene:LOOC260_100080;Name=rpsF;biotype=protein_coding;description=30S ribosomal protein S6;gene_id=LOOC260_		
7	ID=gene:LOOC260_100100;Name=rpsR;biotype=protein_coding;description=30S ribosomal protein S18;gene_id=LOOC260		
8	ID=gene:LOOC260_100290;Name=carB;biotype=protein_coding;description=carbamoyl-phosphate synthase large subunit;		
9	ID=gene:LOOC260_100300;Name=carA;biotype=protein_coding;description=carbamoyl-phosphate synthase small subunit		
10	ID=gene:LOOC260_100310;Name=argC;biotype=protein_coding;description=N-acetyl-gamma-glutamyl-phosphate reducta		
11	ID=gene:LOOC260_100320;Name=argJ;biotype=protein_coding;description=N-acetylglutamate synthase;gene_id=LOOC26		
12	ID=gene:LOOC260_100330;Name=argB;biotype=protein_coding;description=acetylglutamate kinase;gene_id=LOOC260_10		
13	ID=gene:LOOC260_100340;Name=argD;biotype=protein_coding;description=acetylornithine aminotransferase;gene_id=LO		
14	ID=gene:LOOC260_100350;Name=argF;biotype=protein_coding;description=ornithine carbamoyltransferase;gene_id=LOO		
15	ID=gene:LOOC260_100510;Name=horA;biotype=protein_coding;description=multidrug transporter HorA homolog;gene_id=		
16	ID=gene:LOOC260_100520;Name=pepIP;biotype=protein_coding;description=proline iminopeptidase;gene_id=LOOC260_1		
17	ID=gene:LOOC260_101520;Name=ubiE;biotype=protein_coding;description=ubiquinone/menaquinone biosynthesis methy		
18	ID=gene:LOOC260_101730;Name=greA;biotype=protein_coding;description=transcription elongation factor GreA;gene_id=		

awkコマンド

①awk(オーク)コマンドを用いて、②ukyo3.txtに対して、③;を区切り文字として分割後、④2番目のフィールドを抽出した結果を⑤ukyo4.txtに保存している。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [ 2:06午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:06午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo.txt
iu@bielinux[mapping_kiso3] cut -f 9 ukyo2.txt > ukyo3.txt
iu@bielinux[mapping_kiso3] awk -F';' '{print $2}' ukyo3.txt > u
kyo4.txt
iu@bielinux[mapping_kiso3] █ [ 2:06午後 ]
```



ukyo4.txt

かなりスッキリしました。ソート(sort)すれば、さらに見やすくなります。

The screenshot shows the Microsoft Excel interface with a spreadsheet titled "ukyo4.txt". The spreadsheet has 18 rows of data in column A, with columns B through N being empty. The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Name=dnaA													
2	Name=dnaN													
3	Name=recF													
4	Name=gyrB													
5	Name=gyrA													
6	Name=rpsF													
7	Name=rpsR													
8	Name=carB													
9	Name=carA													
10	Name=argC													
11	Name=argJ													
12	Name=argB													
13	Name=argD													
14	Name=argF													
15	Name=horA													
16	Name=pepIP													
17	Name=ubiE													
18	Name=preA													

The Excel interface includes a ribbon with tabs for "ファイル", "ホーム", "挿入", "ページレイアウト", "数式", "データ", "校閲", and "表示". The status bar at the bottom shows "準備完了" and a zoom level of 100%.

状況の整理

今取り扱っているukyo4.txtの、①最初の4行分を表示。②389行です。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [ 2:19午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:19午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo.txt
iu@bielinux[mapping_kiso3] cut -f 9 ukyo2.txt > ukyo3.txt
iu@bielinux[mapping_kiso3] awk -F';' '{print $2}' ukyo3.txt > u
kyo4.txt
① iu@bielinux[mapping_kiso3] head -n 4 ukyo4.txt [ 2:19午後 ]
Name=dnaA
Name=dnaN
Name=recF
Name=gyrB
② iu@bielinux[mapping_kiso3] wc ukyo4.txt [ 2:19午後 ]
389 389 3862 ukyo4.txt
iu@bielinux[mapping_kiso3] █ [ 2:19午後 ]
```

sort

①ukyo4.txtの中身をソートした結果をukyo5.txtに保存。②ukyo5.txtの最初の6行分を表示。確かにアルファベット順にソートできてますね。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [ 2:24午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:24午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo.txt
① iu@bielinux[mapping_kiso3] sort ukyo4.txt > ukyo5.txt
iu@bielinux[mapping_kiso3] head -n 6 ukyo5.txt [ 2:24午後 ]
Name=accA
Name=accB
Name=accC
Name=accD
Name=ackA
Name=acpS
iu@bielinux[mapping_kiso3] [ 2:24午後 ]
```

sort -u

①ソート時に、②重複を除去するオプションをつけて ukyo4.txtをソートした結果の、③行数を表示。365行となりました。これでQuasRのカウント情報取得時のポリシーや手順を(ほぼ)完全に掌握できたこととなります。

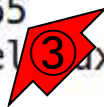
```
File Edit View Search Terminal Help
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:24午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo.txt
iu@bielinux[mapping_kiso3] sort ukyo4.txt > ukyo5.txt
iu@bielinux[mapping_kiso3] head -n 6 ukyo5.txt [ 2:24午後 ]
Name=accA
Name=accB
Name=accC
Name=accD
Name=ackA
Name=acpS
iu@bielinux[mapping_kiso3] sort -u ukyo4.txt | wc [ 2:24午後 ]
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:28午後 ]
```



一行で書くと...

①パイプで連結しまくって一行で書くとこんな感じになります。②GFF3ファイルを入力として、③最終的に365行という結果がちゃんと得られていることがわかります。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [ 2:34午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:34午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo5.txt
ukyo.txt
① iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
| grep -v "tRNA_gene" | grep -v "rRNA_gene" | cut -f 9 | awk -
F';' '{print $2}' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:37午後 ]
③
```



①受講生の長部 高之さん提供コード。このコードをベースとすることで、QuasRの挙動の理解につながりました。

他のやり方



```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3]
iu@bielinux[mapping_kiso3] pwd [ 2:34午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:34午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo5.txt
ukyo.txt
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
| grep -v "tRNA_gene" | grep -v "rRNA_gene" | cut -f 9 | awk -
F';' '{print $2}' | sort -u | wc
365 365 3628
① iu@bielinux[mapping_kiso3] cat *.gff3 | grep -v '^#' | awk '$3
~/gene/{print $9}' | awk -F';' '$2 ~/Name={print $2}' | grep
-v 'L00C' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:43午後 ]
```

説明

①gff3ファイルの中身をパイプで渡し、②#から始まる行を除き…

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3] [ 2:34午後 ]
iu@bielinux[mapping_kiso3] pwd [ 2:34午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:34午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo5.txt
ukyo.txt
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
| grep -v "tRNA_gene" | grep -v "rRNA_gene" | cut -f 9 | awk -
F';' '{print $2}' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] cat *.gff3 | grep -v '^#' | awk '$3
~/gene/{print $9}' | awk -F';' '$2 ~/Name={print $2}' | grep
-v 'L00C' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:43午後 ]
```



説明

③第3フィールド(3列目のこと)に、④geneを含む行の、⑤第9フィールドを抽出。このあたりは非常に美しい記述ですね。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3] [ 2:34午後 ]
iu@bielinux[mapping_kiso3] pwd [ 2:34午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:34午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo5.txt
ukyo.txt
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
| grep -v "tRNA_gene" | grep -v "rRNA_gene" | cut -f 9 | awk -
F';' '{print $2}' | sort -u | wc [ 2:43午後 ]
365 365 3628
iu@bielinux[mapping_kiso3] cat *.gff3 | grep -v '^#' | awk '$3
~/gene/{print $9}' | awk -F';' '$2 ~/Name={print $2}' | grep
-v 'C' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:43午後 ]
```



説明

⑥区切り文字を;として分割し、⑦分割後の第2フィールドに、⑧Name=を含むもののみ、⑨その第2フィールドを抽出。ここまででName=がないものは振り落とされる。

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3] [ 2:34午後 ]
iu@bielinux[mapping_kiso3] pwd [ 2:34午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:34午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo5.txt
ukyo.txt
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
| grep -v "tRNA_gene" | grep -v "rRNA_gene" | cut -f 9 | awk -
F';' '{print $2}' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] cat *.gff3 | grep -v '^#' | awk '$3
~/gene/{print $9}' | awk -F';' '$2 ~/Name={print $2}' | grep
-v 'L00C' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:43午後 ]
```



⑩ LOOC (tRNA_gene や rRNA_gene に相当) を含まないもののみ抽出して、⑪ 重複除去後に行数をカウント

説明

```
iu@bielinux[~/Desktop/mac_share/mapping_kiso3] [ 2:34午後 ]
iu@bielinux[mapping_kiso3] pwd [ 2:34午後 ]
/home/iu/Desktop/mac_share/mapping_kiso3
iu@bielinux[mapping_kiso3] ls [ 2:34午後 ]
Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chrom
osome.Chromosome.gff3
uge.txt
ukyo2.txt
ukyo3.txt
ukyo4.txt
ukyo5.txt
ukyo.txt
iu@bielinux[mapping_kiso3] grep "ID=gene" *.gff3 | grep "Name="
| grep -v "tRNA_gene" | grep -v "rRNA_gene" | cut -f 9 | awk -
F';' '{print $2}' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] cat *.gff3 | grep -v '^#' | awk '$3
~/gene/{print $9}' | awk -F';' '$2 ~/Name={print $2}' | grep
-v 'LOOC' | sort -u | wc
365 365 3628
iu@bielinux[mapping_kiso3] [ 2:43午後 ]
```

⑩

⑪

おさらい

①のsingle-endでアノテーション有の、②例題10の実行結果として、365遺伝子のカウントデータしか得られなかった。機能ゲノム学第4回のスライド82-99

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページはフリーソフトRとWindows2015 (2015/04/03)

What's new?

- 「マップ後」
- 「イントロ」
- 「H29年度N

- マップ後 | 出力ファイルの読み込み | [htSeqTools\(Planet 2012\)](#) (last modified 2013/06/19)
- マップ後 | [カウント情報取得](#) | [|](#)について (last modified 2018/05/30) **NEW**
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/29) **①**
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [HTSeq\(Anders 2015\)](#) (last modified 2018/05/30) **N**
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション無 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/26)
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2016/02/12)

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR (Gaidatzis_2015) **NEW**

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カウントデータ取得までの一連の流れを示します。アノテーション情報は、GenomicFeaturesパッケージ中の関数を利用してTxDbオブジェクトをネットワーク経由で取得するのを基本としつつ、TxDbパッケージを読み込むやり方も示しています。マッピングのやり方やアノテーションの詳細についてはマッピング(single-endゲノム | アノテーション有) | QuasR(Gaidatzis_2015)などを参考にしてください。



② 10.mapping single genome7.txt中のFASTA形式ファイルを乳酸菌ゲノムにマッピングする場合:

マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl(Zerbino et al., Nucleic Acids Res., 2018)から提供されているLactobacillus casei 12Aのmulti-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。マッピング結果に対して、GFF3形式のアノテーションファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3)を読み込んでカウント情報を取得しています。

1. サンプル
合:
mapping
目の2列目
-best -str
して、UC
in_f1 <
in_f2 <
out_f <
param_n

```
in_f1 <- "mapping_single_genome7.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome
in_f3 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff
out_f <- "hoge10.txt" #出力ファイル名を指定してout_fに格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定: "gene", "exon", "pro

#必要なパッケージをロード
library(QuasR) #パッケージの読み込み
library(GenomicFeatures) #パッケージの読み込み
```

QuasRのカウント情報は

①tRNAやrRNAの結果は含まれない。② genenameがないものも含まれない。③同じgenenameのものは重複除去されている。④(デフォルトでは)領域内にマップされたリードのみカウントしている。

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ Vignettes

R Console
> tmp <- cbind(rownames(count), count) #保存し$
> write.table(tmp, out_f, sep="\t", append=F, qu$
> getwd()
[1] "C:/Users/kojik/Desktop/hoge/mapping_kiso3"
> list.files()
[1] "hoge10.txt" $
[2] "Lactobacillus_hokkaidonensis_jcm_18461.GCA$
[3] "Lactobacillus_hokkaidonensis_jcm_18461.GCA$
[4] "Lactobacillus_hokkaidonensis_jcm_18461.GCA$
[5] "Lactobacillus_hokkaidonensis_jcm_18461.GCA$
[6] "Lactobacillus_hokkaidonensis_jcm_18461.GCA$
[7] "mapping_single_genome7.txt" $
[8] "QuasR_log_3b6c12e745f9.txt" $
[9] "sample_RNAseq4.fa" $
[10] "sample_RNAseq4_3b6c652a602a.bam" $
[11] "sample_RNAseq4_3b6c652a602a.bam.bai" $
[12] "sample_RNAseq4_3b6c652a602a.bam.txt" $
> |
```

	width	Lacto
accA	750	0
accB	369	0
accC	1347	0
accD	789	0
ackA	1191	0
acpS	363	0
addA	3711	0
adh	1056	0
adhE	2607	0
adk	660	0
alaS	2655	0
aldA	1464	0
aldB	714	0
amt	1323	0

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

HTSeq

①HTSeqは、2015年の出版以降②PubMedのみで2,000回以上引用されている有名なPythonで書かれたプログラム。ここでは、HTSeqに含まれる③htseq-countを利用してカウント情報の取得を行う。

Bioinformatics. 2015 Jan 15;31(2):166-9. doi: 10.1093/bioinformatics/btu638. Epub 2014 Sep 25.

HTSeq--a Python framework to work with high-throughput sequencing data.

Anders S¹, Pyl PT¹, Huber W¹.

Author information

Abstract

MOTIVATION: A large choice of tools exists for many standard tasks in the analysis of high-throughput sequencing (HTS) data. However, once a project deviates from standard workflows, custom scripts are needed.

RESULTS: We present HTSeq, a Python library to facilitate the rapid development of such scripts. HTSeq offers parsers for many common data formats in HTS projects, as well as classes to represent data, such as genomic coordinates, sequences, sequencing reads, alignments, gene model information and variant calls, and provides data structures that allow for querying via genomic coordinates. We also present htseq-count, a tool developed with HTSeq that preprocesses RNA-Seq data for differential expression analysis by counting the overlap of reads with genes.

AVAILABILITY AND IMPLEMENTATION: HTSeq is released as an open-source software under the GNU General Public Licence and available from <http://www-huber.embl.de/HTSeq> or from the Python Package Index at <https://pypi.python.org/pypi/HTSeq>.

© The Author 2014. Published by Oxford University Press.

PMID: 25260700 PMCID: [PMC4287950](#) DOI: [10.1093/bioinformatics/btu638](#)

[Indexed for MEDLINE] [Free PMC Article](#)



Images from this publication. [See all images \(2\)](#) [Free text](#)



PMC **FREE** Full text

Save items

★ Add to Favorites

Similar articles

Rcount: simple and flexible RNA-Seq read counting. [Bioinformatics. 2015]

omics Pipe: a community-based framework for re [Bioinformatics. 2015]

TSSV: a tool for characterization of complex allelic v [Bioinformatics. 2014]

Review Toward better understanding of artifacts in var [Bioinformatics. 2014]

Review Bioinformatics tools for analysing viral ge [Rev Sci Tech. 2016]

[See reviews...](#)

[See all...](#)

Cited by over 100 PubMed Central articles

Intron retention and nuclear loss of SFPQ are molec [Nat Commun. 2018]

Transcriptome analysis of the adult human Klinefelter [Cell Death Dis. 2018]

Regulation Mechanism Mediated by

HTSeq

講義日程 (平成30年度)

1. 平成30年06月12日 (PC使用)

講義資料PDF

QuasR : Gaidatzis et al., Bioinf

HTSeq : Anders et al., Bioinform



The screenshot shows a web browser window displaying the HTSeq 0.10.0 documentation page. The browser's address bar shows the URL `http://htseq.readthedocs.io/en/release_0.10.0/`. The page title is "HTSeq: Analysing high-throughput sequencing data with Python". The main content area contains a table of contents with the following items:

- Overview
 - Paper
 - Documentation overview
 - Author
 - License
- Prerequisites and installation
 - Installation on Linux
 - Installation on MacOS X
 - MS Windows
- A tour through HTSeq
 - Reading in reads
 - Reading and writing BAM files
 - Genomic intervals and genomic arrays
 - Counting reads by genes
 - And much more
- A detailed use case: TSS plots
 - Using the full coverage
 - Using indexed BAM files
 - Streaming through all reads
- Counting reads
 - Preparing the feature array
 - Counting ungapped single-end reads

At the bottom right of the page, there is a dropdown menu showing "v. release_0.10.0".

Two red arrows are overlaid on the screenshot: Arrow ① points to the "HTSeq : Anders et al., Bioinform" link in the lecture schedule on the left. Arrow ② points to the bottom right corner of the browser window, indicating the scroll action.



HTSeq

講義日程 (平成30年度)

1. 平成30年06月12日 (PC使用)

講義資料PDF

QuasR : Gaidatzis et al., Bioinformatics

HTSeq : Anders et al., Bioinformatics

The screenshot shows a web browser window with the URL `http://htseq.readthedocs.io/en/release_0.10.0/`. The page content is a navigation menu with the following items:

- Multiple alignments
- CIGAR strings
- Features
 - `GFF_Reader` and `GenomicFeature`
- Other parsers
 - `VCF_Reader` and `VariantCall`
 - Wiggle Reader
 - BED Reader
- Miscellaneous
 - `FileOrSequence`
 - Version
- Quality Assessment with `htseq-qa`
 - Plot
 - Usage
- Counting reads in features with `htseq-count`
 - Usage
- Version history
 - Version 0.10.0
 - Version 0.9.1
 - Version 0.9.0
 - Version 0.8.0
 - Version 0.7.2
 - Version 0.7.1
 - Version 0.7.0
 - Version 0.6.1
 - Version 0.6.0
 - Version 0.5.4
 - Version 0.5.3
 - Version 0.5.2

At the bottom right of the page, there is a dropdown menu showing `v. release_0.10.0`. Two red arrows with circled numbers are overlaid on the image: arrow 1 points to the 'v. release_0.10.0' dropdown, and arrow 2 points to the 'htseq-count' menu item.

htseq-count

① htseq-count のページ。ここでは、htseq-count が提供する② 3つのカウントモードや、どのようにオプションを使いこなして自分の欲しいカウントデータを得るのかについて解説。

The screenshot shows the web browser interface for the htseq-count documentation. The address bar shows the URL: `http://htseq.readthedocs.io/en/release_0.10.0/count.html`. The page title is "Counting reads in features with htseq-count". A red arrow labeled "1" points to the page title. The left sidebar contains a "Table of Contents" with links to "Usage", "Options", and "Frequently asked questions". Below that are "Previous topic" (Quality Assessment with htseq-qa) and "Next topic" (Version history). The "This Page" section has a "Show Source" link. A "Quick search" box is at the bottom of the sidebar. The main content area has a dark blue header with "previous | next | index" links. The main text starts with "Given a file with aligned sequencing reads and a list of genomic features, a common task is to count how many reads map to each feature." It then explains that a feature is an interval on a chromosome. It mentions that in RNA-Seq, features are typically genes. It notes that the htseq-count script allows choosing between three modes. A red arrow labeled "2" points to the sentence: "The htseq-count script allows to choose between three modes." Below this, it says "Of course, if none of these fits your needs, you can write your own script with HTSeq." It then explains the three overlap resolution modes: union, intersection, and intersection-strict. A dropdown menu at the bottom right shows "v: release_0.10.0".

HTSeq 0.10.0 documentation » [previous](#) | [next](#) | [index](#)

Counting reads in features with htseq-count

Given a file with aligned sequencing reads and a list of genomic features, a common task is to count how many reads map to each feature.

A feature is here an interval (i.e., a range of positions) on a chromosome or a union of such intervals.

In the case of RNA-Seq, the features are typically genes, where each gene is considered here as the union of all its exons. One may also consider each exon as a feature, e.g., in order to check for alternative splicing. For comparative ChIP-Seq, the features might be binding region from a pre-determined list.

Special care must be taken to decide how to deal with reads that align to or overlap with more than one feature. The `htseq-count` script allows to choose between three modes. Of course, if none of these fits your needs, you can write your own script with HTSeq. See the chapter [A tour through HTSeq](#) for a step-by-step guide on how to do so. See also the FAQ at the end, if the following explanation seems too technical.

The three overlap resolution modes of `htseq-count` work as follows. For each position i in the read, a set $S(i)$ is defined as the set of all features overlapping position i . Then, consider the set S , which is (with i running through all position within the read or a read pair)

- the union of all the sets $S(i)$ for mode `union`. This mode is recommended for most use cases.
- the intersection of all the sets $S(i)$ for mode `intersection-strict`.

3つのカウントモード

①ちょっと下に移動。②3つのカウントモードに関する説明。赤下線部分が指定するオプション名で、③unionが推奨のようだ。この段階でデフォルトはunionだと判断する。

The screenshot shows a web browser displaying the HTSeq documentation page for counting reads in features. The page content includes:

Of course, if none of these fits your needs, you can write your own script with HTSeq. See the chapter [A tour through HTSeq](#) for a step-by-step guide on how to do so. See also the FAQ at the end, if the following explanation seems too technical.

The three overlap resolution modes of `htseq-count` work as follows. For each position i in the read, a set $S(i)$ is defined as the set of all features overlapping position i . Then, consider the set S , which is (with i running through all position within the read or a read pair)

- the union of all the sets $S(i)$ for mode union. This mode is recommended for most use cases.
- the intersection of all the sets $S(i)$ for mode intersection-strict.
- the intersection of all non-empty sets $S(i)$ for mode intersection-nonempty.

If S contains precisely one feature, the read (or read pair) is counted for this feature. If S is empty, the read (or read pair) is counted as `no_feature`. If S contains more than one feature, `htseq-count` behaves differently based on the `--nonunique` option:

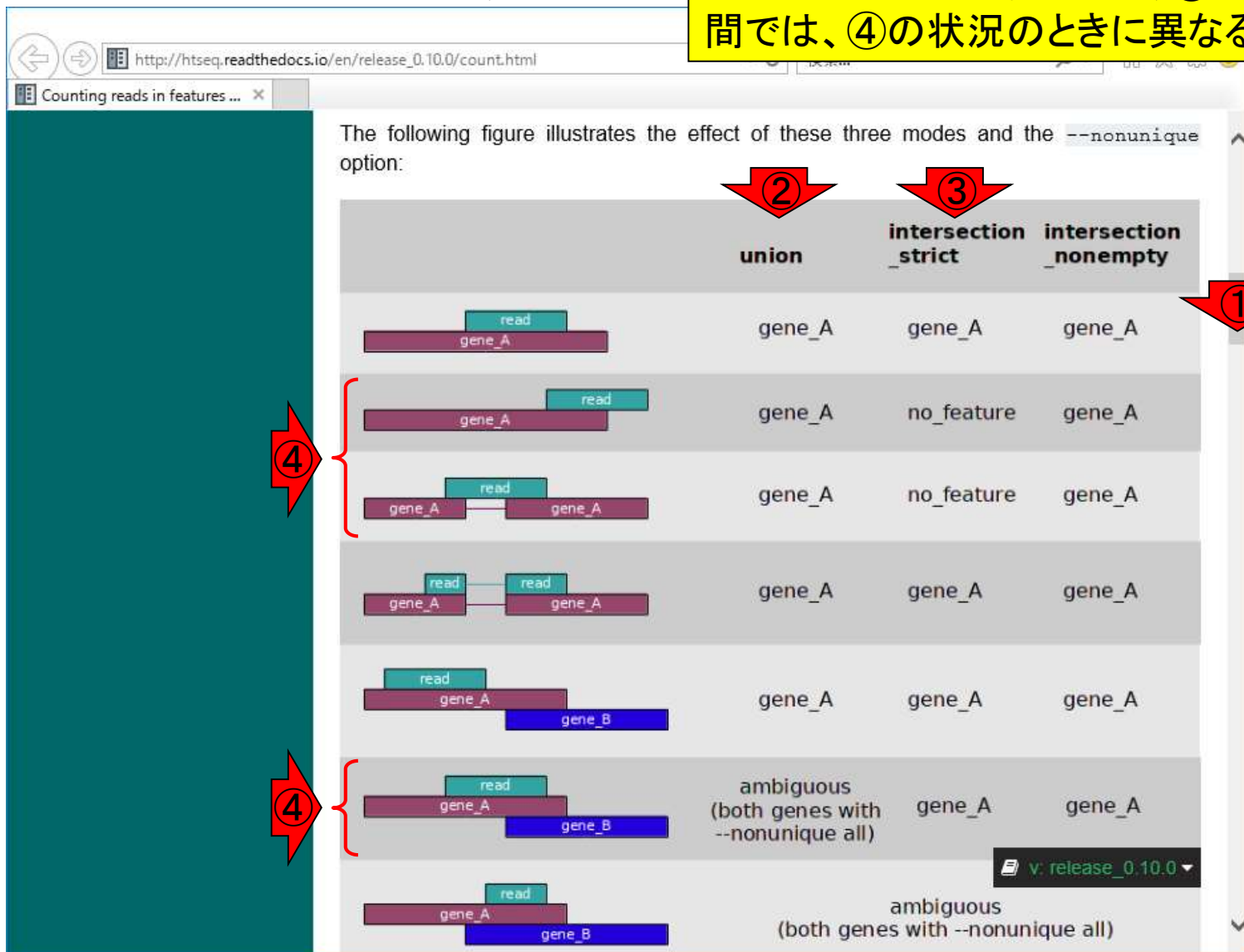
- `--nonunique none` (default): the read (or read pair) is counted as `ambiguous` and not counted for any features. Also, if the read (or read pair) aligns to more than one location in the reference, it is scored as `alignment_not_unique`.
- `--nonunique all`: the read (or read pair) is counted as `ambiguous` and is also counted in all features to which it was assigned. Also, if the read (or read pair) aligns to more than one location in the reference, it is scored as `alignment_not_unique` and also separately for each location.

Notice that when using `--nonunique all` the sum of all counts will be the number of reads (or read pairs), because those with multiple alignments are scored multiple times.

Annotations on the screenshot: A red arrow labeled '2' points to the list of three modes. A red arrow labeled '1' points to the text 'The three overlap resolution modes...'. A red arrow labeled '3' points to the 'union' mode in the list.

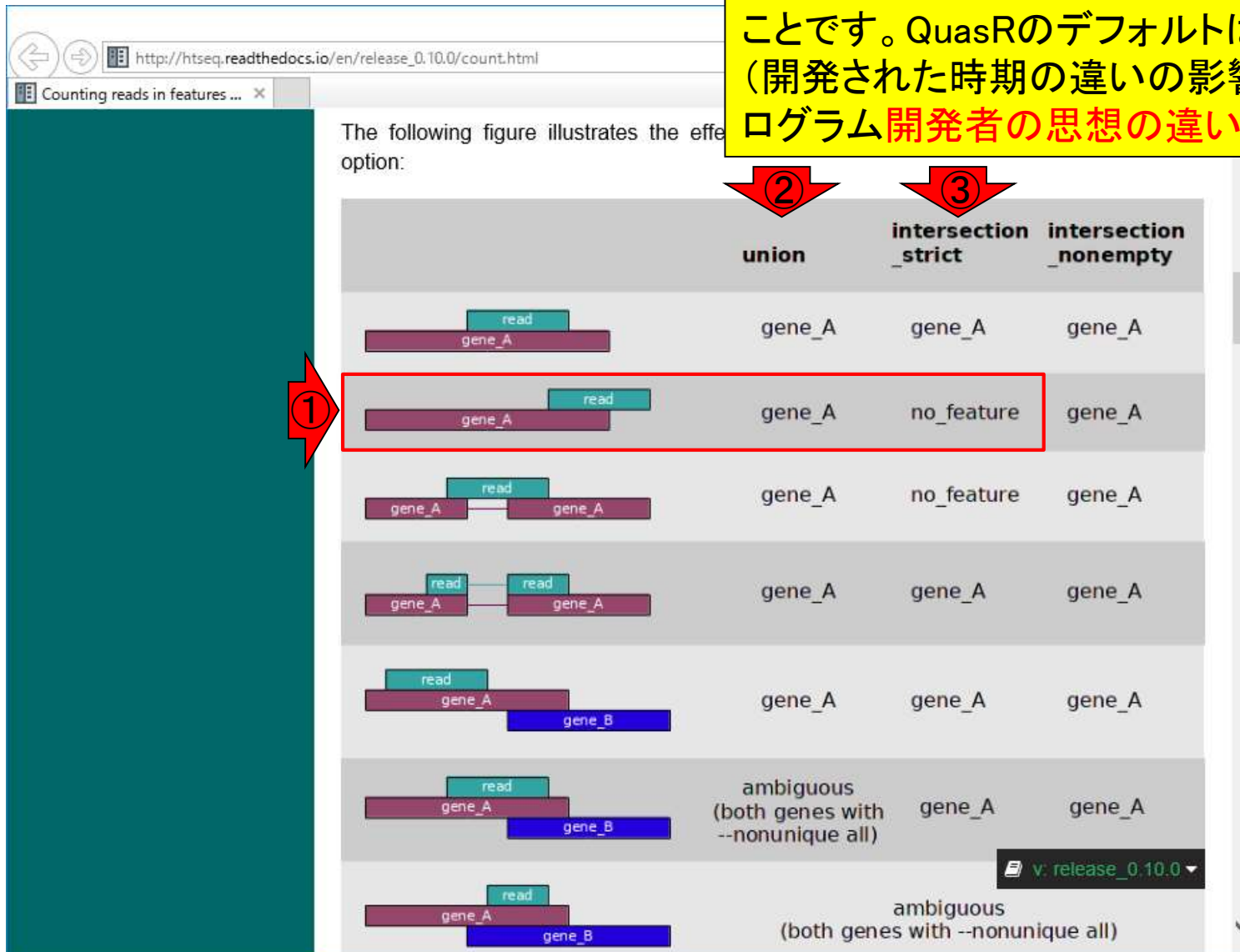
モード間の違い

①ちょっと下に移動。リードがgene_A上にマップされたと判断する基準について、どのモードを選択するとどう判定されるかが示されている。例えば、②unionと③intersection_strict間では、④の状況のときに異なる結果になる。



QuasRとの違い

①はgene_Aの領域にリードの一部がマップされている状況です。②デフォルトのunionモードではリードをカウントするが、③intersection_strictモードではカウントしないということです。QuasRのデフォルトはカウントしない、でしたね。(開発された時期の違いの影響ももちろんありますが)プログラム開発者の思想の違いがわかる例です。



おさらい

①のsingle-endでアノテーション有の、②例題10の実行結果として、365遺伝子のカウントデータしか得られなかった。機能ゲノム学第4回のスライド82-99。③出力ファイル(hoge10.txt)の中身が…

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページは、フリーソフトRとWindows2015 (2015/04/03)

What's new?

- 「マップ後」
- 「イントロ」
- 「H29年度N

- マップ後 | 出力ファイルの読み込み | [htSeqTools\(Planet 2012\)](#) (last modified 2013/06/19)
- マップ後 | [カウント情報取得](#) | [|について](#) (last modified 2018/05/30) **NEW**
- マップ後 | [カウント情報取得](#) | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) **①** (modified 2018/05/29)
- マップ後 | [カウント情報取得](#) | single-end | ゲノム | アノテーション有 | [HTSeq\(Anders 2015\)](#) (last modified 2018/05/30) **N**
- マップ後 | [カウント情報取得](#) | single-end | ゲノム | アノテーション無 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/26)
- マップ後 | [カウント情報取得](#) | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2016/02/12)

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR (Gaidatzis_2015) **NEW**

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カウントデータ取得までの一連の流れを示します。アノテーション情報は、GenomicFeaturesパッケージ中の関数を利用してTxDbオブジェクトをネットワーク経由で取得するのを基本としつつ、TxDbパッケージを読み込むやり方も示しています。マッピングのやり方やアノテーションの詳細については、[マッピング\(single-end\) | ゲノム | アノテーション\(応用\) | QuasR\(Gaidatzis 2015\)](#)などを参考にしてください。



② **10.mapping single genome7.txt**中のFASTA形式ファイルを乳酸菌ゲノムにマッピングする場合:
マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl \(Zerbino et al., Nucleic Acids Res., 2018\)](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。マッピング結果に対して、GFF3形式のアノテーションファイル ([Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#))を読み込んでカウント情報を取得しています。



```
in_f1 <- "mapping_single_genome7.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome
in_f3 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff
out_f <- "hoge10.txt" #出力ファイル名を指定してout_fに格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定: "gene", "exon", "pro

#必要なパッケージをロード
library(QuasR) #パッケージの読み込み
library(GenomicFeatures) #パッケージの読み込み
```


おさらい

① hoge.10.txtの中身。ヘッダー(行名や列名)部分を除く、②行数と列数は365と2。②全365遺伝子の領域にマップされたリードの総数(カウント総数)は、③2でした。

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ
[Icons]
R Console
> dim(count)
[1] 365 2
> head(count)
      width Lacto
accA    750     0
accB    369     0
accC   1347     0
accD    789     0
ackA   1191     0
acpS    363     0
> head(count[,2])
accA accB accC accD ackA acpS
  0    0    0    0    0    0
> sum(count[,2])
[1] 2
> |
```

①

	width	Lacto
accA	750	0
accB	369	0
accC	1347	0
accD	789	0
ackA	1191	0
acpS	363	0
addA	3711	0
adh	1056	0
adhE	2607	0
adk	660	0
alaS	2655	0
aldA	1464	0
aldB	714	0
amt	1323	0

おさらい

②

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
Chromosome ena gene 360 1676 . + ID=ge
Chromosome ena transcript 360 1676 . + ID=tr
Chromosome ena exon 360 1676 . + Pare
Chromosome ena CDS 360 1676 . + 0 ID=C
###
Chromosome ena gene 1852 2991 . + ID=ge
Chromosome ena transcript 1852 2991 . + ID=tr
Chromosome ena exon 1852 2991 . + Pare
Chromosome ena CDS 1852 2991 . + 0 ID=C
###
Chromosome ena gene 3233 3457 . + ID=ge
Chromosome ena transcript 3233 3457 . + ID=tr
Chromosome ena exon 3233 3457 . + Pare
Chromosome ena CDS 3233 3457 . + 0 ID=C
###
Chromosome ena gene 3467 4588 . + ID=ge
```

③

③

③

③

①マップする側 (sample_RNAseq4.fa) の計 11リードは、②アノテーションファイル (.gff3) 中の③gene領域を参考にしながら作成。 全て完全一致でマップされるように設計。

①

```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAGAAC
```

①最初の2リードは、②の領域内にマップされるように設計。これがQuasRのカウント総数2に相当する。

おさらい

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
```

Chromosome	ena	gene	360	1676	.	+	.	ID=ge
Chromosome	ena	transcript	360	1676	.	+	.	ID=tr
Chromosome	ena	exon	360	1676	.	+	.	Pare
Chromosome	ena	CDS	360	1676	.	+	0	ID=C
###								
Chromosome	ena	gene	1852	2991	.	+	.	ID=ge
Chromosome	ena	transcript	1852	2991	.	+	.	ID=tr
Chromosome	ena	exon	1852	2991	.	+	.	Pare
Chromosome	ena	CDS	1852	2991	.	+	0	ID=C
###								
Chromosome	ena	gene	3233	3457	.	+	.	ID=ge
Chromosome	ena	transcript	3233	3457	.	+	.	ID=tr
Chromosome	ena	exon	3233	3457	.	+	.	Pare
Chromosome	ena	CDS	3233	3457	.	+	0	ID=C
###								
Chromosome	ena	gene	3467	4588	.	+	.	ID=ge

①

```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
```

②

```
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAAGAAC
```

おさらい

①3-7番目の5リードは、②の領域に一部がかかるように設計。領域内ではないため、QuasRではカウントされないが、HTSeqではカウントされると予想。

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
```

Chromosome	ena	gene	360	1676	+	ID=ge
Chromosome	ena	transcript	360	1676	+	ID=tr
Chromosome	ena	exon	360	1676	+	Pare
Chromosome	ena	CDS	360	1676	+	0 ID=C
###						
Chromosome	ena	gene	1852	2991	+	ID=ge
Chromosome	ena	transcript	1852	2991	+	ID=tr
Chromosome	ena	exon	1852	2991	+	Pare
Chromosome	ena	CDS	1852	2991	+	0 ID=C
###						
Chromosome	ena	gene	3233	3457	+	ID=ge
Chromosome	ena	transcript	3233	3457	+	ID=tr
Chromosome	ena	exon	3233	3457	+	Pare
Chromosome	ena	CDS	3233	3457	+	0 ID=C
###						
Chromosome	ena	gene	3467	4588	+	ID=ge



```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAGAAC
```


おさらい

①8番目のリードは、②の領域内にマップされるように設計。QuasRではgenenameがなかったため、計365遺伝子からなるカウントデータには反映されなかった。HTSeqではどのような取り扱いになるのだろうか？

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
```

Chromosome	ena	gene	360	1676	.	+	.	ID=ge
Chromosome	ena	transcript	360	1676	.	+	.	ID=tr
Chromosome	ena	exon	360	1676	.	+	.	Pare
Chromosome	ena	CDS	360	1676	.	+	0	ID=C
###								
Chromosome	ena	gene	1852	2991	.	+	.	ID=ge
Chromosome	ena	transcript	1852	2991	.	+	.	ID=tr
Chromosome	ena	exon	1852	2991	.	+	.	Pare
Chromosome	ena	CDS	1852	2991	.	+	0	ID=C
###								
Chromosome	ena	gene	3233	3457	.	+	.	ID=ge
Chromosome	ena	transcript	3233	3457	.	+	.	ID=tr
Chromosome	ena	exon	3233	3457	.	+	.	Pare
Chromosome	ena	CDS	3233	3457	.	+	0	ID=C
###								
Chromosome	ena	gene	3467	4588	.	+	.	ID=ge

```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAGAAC
```



おさらい

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
```

Chromosome	ena	gene	360	1676	.	+	.	ID=ge
Chromosome	ena	transcript	360	1676	.	+	.	ID=tr
Chromosome	ena	exon	360	1676	.	+	.	Pare
Chromosome	ena	CDS	360	1676	.	+	0	ID=C
###								
Chromosome	ena	gene	1852	2991	.	+	.	ID=ge
Chromosome	ena	transcript	1852	2991	.	+	.	ID=tr
Chromosome	ena	exon	1852	2991	.	+	.	Pare
Chromosome	ena	CDS	1852	2991	.	+	0	ID=C
###								
Chromosome	ena	gene	3233	3457	.	+	.	ID=ge
Chromosome	ena	transcript	3233	3457	.	+	.	ID=tr
Chromosome	ena	exon	3233	3457	.	+	.	Pare
Chromosome	ena	CDS	3233	3457	.	+	0	ID=C
###								
Chromosome	ena	gene	3467	4588	.	+	.	ID=ge

①9-10番目のリードは、②の領域に一部がかかるように設計。領域内ではない。QuasRではgenenameがなかったため、計365遺伝子からなるカウントデータには反映されなかった。HTSeqで取り扱われるのであれば、この2リードもカウントされるだろうと予想。

```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAGAAC
```



おさらい

①11番目のリードは、②と③の領域にまたがってマップされるように設計。HTSeq(のデフォルトであるunionモード)では、ambiguousとなるであろう。

##gff-version 3						
##sequence-region Chromosome 360 2277853						
#!genome-build European Nucleotide Archive ASM82939						
#!genome-version GCA_000829395.1						
#!genome-date 2014-11						
#!genome-build-accession GCA_000829395.1						
#!genebuild-last-updated 2014-11						
Chromosome	ena	gene	360	1676	+	ID=ge
Chromosome	ena	transcript	360	1676	+	ID=tr
Chromosome	ena	exon	360	1676	+	Pare
Chromosome	ena	CDS	360	1676	+	0 ID=C
###						
Chromosome	ena	gene	1852	2991	+	ID=ge
Chromosome	ena	transcript	1852	2991	+	ID=tr
Chromosome	ena	exon	1852	2991	+	Pare
Chromosome	ena	CDS	1852	2991	+	0 ID=C
###						
Chromosome	ena	gene	3233	3457	+	ID=ge
Chromosome	ena	transcript	3233	3457	+	ID=tr
Chromosome	ena	exon	3233	3457	+	Pare
Chromosome	ena	CDS	3233	3457	+	0 ID=C
###						
Chromosome	ena	gene	3467	4588	+	ID=ge

```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAGAAC
```



ambiguous

HTSeq(のデフォルトである①unionモード)では、②複数の領域にまたがってマップされるリードは、ambiguousとなる。

The following figure illustrates the effect of these three modes and the `--nonunique` option:

	union	intersection_strict	intersection_nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous (both genes with --nonunique all)	gene_A	gene_A
	ambiguous (both genes with --nonunique all)	ambiguous	ambiguous (both genes with --nonunique all)



Contents

■ カウント情報取得の続き

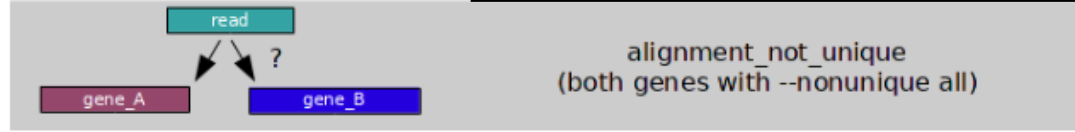
- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

Usage (利用法)

Linux用プログラムでよく見かける①Usageの見方について説明します。②の[...]や<...>の違いから始めます。Linuxではなく、クラウド解析環境(DDBJ PipelineやGalaxy)を利用する場合も無関係ではない。オプションの詳細については、ここで示すような大元のプログラムのマニュアルページを読み解く必要がある場合があります。



① Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

```
htseq-count [options] <alignment_files> <gff_file>
```

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```

The `<alignment_files>` are one or more files containing the aligned reads in SAM format. ([SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as [STAR](#). HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

The `<gff_file>` contains the features in the GFF format.

v. release_0.10.0

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

[options]はなくてもよい

① [options]は、文字通りオプションなので、実行時になくてもよいものです。

Counting reads in features ... x

http://htseq.readthedocs.io/en/release_0.10.0/count.html 検索...

read

gene_A gene_B

alignment_not_unique
(both genes with --nonunique all)

Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

```
htseq-count [options] <alignment_files> <gff_file>
```

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```

The `<alignment_files>` are one or more files containing the aligned reads in SAM format. ([SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as [STAR](#). HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

The `<gff_file>` contains the features in the [GFF format](#).

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

v. release_0.10.0

<...>は絶対必要

①<alignment_files>や②<gff_file>は、③htseq-countプログラムの実行時に必須のファイル達です。確かに①マッピング結果ファイルや、②アノテーションファイルがないとカウント情報を取得しようがないので妥当ですね。

Counting reads in features ... x

http://htseq.readthedocs.io/en/release_0.10.0/count.html

read

gene_A gene_B

alignment_not_unique
(both genes with --nonunique all)

Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

```
htseq-count [options] <alignment_files> <gff_file>
```

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```

The `<alignment_files>` are one or more files containing the aligned reads in SAM format. ([SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as [STAR](#). HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

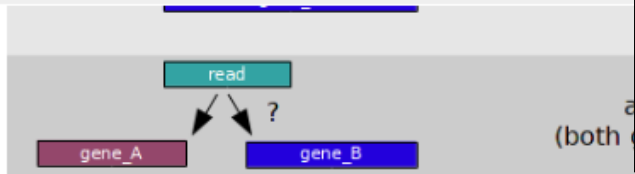
The `<gff_file>` contains the features in the GFF format.

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

v. release_0.10.0

<alignment_files>

(複数ファイルを一度に実行したいヒトは)①
<alignment_files>と複数形になっているのを見逃さない。②でもSAM形式であれば、複数ファイルを指定可能と書いてある。BAMファイル(SAMのバイナリ版)とは書いてはいないものの、2015年に出版された論文でBAMを取り扱えないことはないでしょうという思想のもと、後にoptionのところではBAMファイルの取り扱いに関する記述を探す。



Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:



```
htseq-count [options] <alignment_files> <gff_file>
```

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```



The `<alignment_files>` are one or more files containing the aligned reads in SAM format. (SAMtools contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as STAR. HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

The `<gff_file>` contains the features in the GFF format.

v. release_0.10.0

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

<gff_file>

アノテーションファイルは複数のわけがないので、①<gff_file>となっているのは妥当ですね。②GFF形式のようです。但し！、③のリンク先がGFF2 (GTF形式のこと)となっていることから、デフォルトはGFF3ではないのだろうと予想を立てる。

Counting reads in features ... x

http://htseq.readthedocs.io/en/release_0.10.0/count.html

read

gene_A gene_B

alignment_not_unique
(both genes with --nonunique all)

Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

```
htseq-count [options] <alignment_files> <gff_file>
```

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```

The `<alignment_files>` are one or more files containing the aligned reads in SAM format. ([SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as [STAR](#). HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

② The `<gff_file>` contains the features in the GFF format.

v. release_0.10.0

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

strand-specific

①少しページ下部に移動。②strand-specific protocolで得られたRNA-seqデータをデフォルトにするのは妥当。そうでないデータの場合は、得られるカウント総数が当然少なくなるので`--stranded=no`オプションをつけねばならないと書かれている。

- `__not_aligned`: reads (or read pairs) in the SAM file without alignment
- `__alignment_not_unique`: reads (or read pairs) with more than one reported alignment. These reads are recognized from the `NH` optional SAM field tag. (If the aligner does not set this field, multiply aligned reads will be counted multiple times, unless they get filtered out by due to the `-a` option.) Note that if the `--nonunique all` option was used, these reads (or read pairs) are still assigned to features.

② *Important:* The default for strandedness is `yes`. If your RNA-Seq data has not been made with a strand-specific protocol, this causes half of the reads to be lost. Hence, make sure to set the option `--stranded=no` unless you have strand-specific data!

Options

`-f <format>`, `--format=<format>`

Format of the input data. Possible values are `sam` (for text SAM files) and `bam` (for binary BAM files). Default is `sam`.

`-r <order>`, `--order=<order>`

For paired-end data, the alignment have to be sorted either by read name or by alignment position. If your data is not sorted, use the `samtools sort` function of `samtools` to sort it. Use this option, with `name` or `pos` for `<order>` to indicate how the input data has been sorted. The default is `name`.

If `name` is indicated, `htseq-count` expects all the alignments for the reads of a given read pair to appear in adjacent records in the input data. For `pos`, this is not expected; rather, read alignments whose mate alignment have not yet been seen are kept in a buffer in memory until the mate is found. While, strictly speaking, this is not a problem with unsorted data, sorting ensures that most alignment mates appear close to each other in the data and hence the buffer is much less likely to overflow.



Options

ここからが①オプションの説明部分。②これはフォーマットの指定に関するもの。マッピング結果ファイルのフォーマットに関するものであることは、その後の記述内容からわかる。デフォルトはSAM形式であるので、SAMファイルを読み込ませる場合は、`-f`や`--format`のオプションは書かなくてもよいことがわかる。BAMファイルを入力として与えたい場合は、`-f bam`や`--format=bam`と書けばよいのだと読み解く。

- `__not_aligned`: reads (or read pairs) that were not aligned by the aligner does not set this field, multiply aligned reads will be counted multiple times, unless they getv filtered out by due to the `-a` option.) Note that if the `--nonunique all` option was used, these reads (or read pairs) are still assigned to features.

Important: The default for strandedness is `yes`. If your RNA-Seq data has not been made with a strand-specific protocol, this causes half of the reads to be lost. Hence, make sure to set the option `--stranded=no` unless you have strand-specific data!

① Options

-f <format>, **--format**=<format>

Format of the input data. Possible values are `sam` (for text SAM files) and `bam` (for binary BAM files). Default is `sam`.

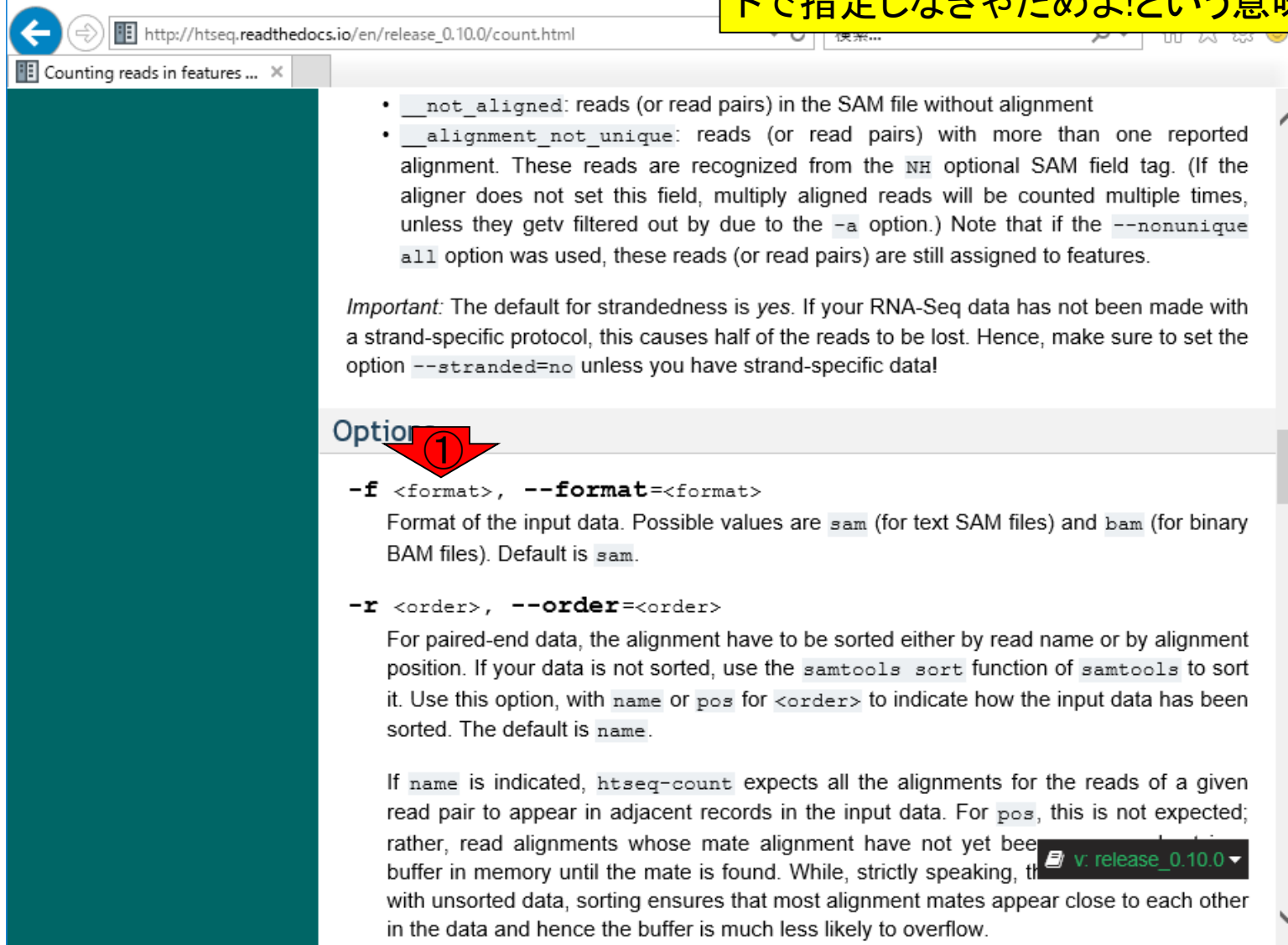
-r <order>, **--order**=<order>

For paired-end data, the alignment have to be sorted either by read name or by alignment position. If your data is not sorted, use the `samtools sort` function of `samtools` to sort it. Use this option, with `name` or `pos` for <order> to indicate how the input data has been sorted. The default is `name`.

If `name` is indicated, `htseq-count` expects all the alignments for the reads of a given read pair to appear in adjacent records in the input data. For `pos`, this is not expected; rather, read alignments whose mate alignment have not yet been read are kept in a buffer in memory until the mate is found. While, strictly speaking, this is not a problem with unsorted data, sorting ensures that most alignment mates appear close to each other in the data and hence the buffer is much less likely to overflow.

Options

①の<format>という記述に注目!。<…>は絶対に必要なもの、という意味でした。これは、`-f`自体はつけてもつけなくてもよいものだが、`-f`をつける場合は`sam`または`bam`もセットで指定しなきゃだめよ!という意味です。



Counting reads in features ... x

http://htseq.readthedocs.io/en/release_0.10.0/count.html

- `__not_aligned`: reads (or read pairs) in the SAM file without alignment
- `__alignment_not_unique`: reads (or read pairs) with more than one reported alignment. These reads are recognized from the `NH` optional SAM field tag. (If the aligner does not set this field, multiply aligned reads will be counted multiple times, unless they getv filtered out by due to the `-a` option.) Note that if the `--nonunique all` option was used, these reads (or read pairs) are still assigned to features.

Important: The default for strandedness is `yes`. If your RNA-Seq data has not been made with a strand-specific protocol, this causes half of the reads to be lost. Hence, make sure to set the option `--stranded=no` unless you have strand-specific data!

Options

①

-f <format>, **--format**=<format>

Format of the input data. Possible values are `sam` (for text SAM files) and `bam` (for binary BAM files). Default is `sam`.

-r <order>, **--order**=<order>

For paired-end data, the alignment have to be sorted either by read name or by alignment position. If your data is not sorted, use the `samtools sort` function of `samtools` to sort it. Use this option, with `name` or `pos` for <order> to indicate how the input data has been sorted. The default is `name`.

If `name` is indicated, `htseq-count` expects all the alignments for the reads of a given read pair to appear in adjacent records in the input data. For `pos`, this is not expected; rather, read alignments whose mate alignment have not yet been seen are kept in a buffer in memory until the mate is found. While, strictly speaking, this is not possible with unsorted data, sorting ensures that most alignment mates appear close to each other in the data and hence the buffer is much less likely to overflow.

strand-specific

①少しページ下部に移動。②がOptionsの説明の少し上の部分に書かれていた、strand-specific protocolで得られたRNA-seqデータの取り扱いに関する詳細な説明。--stranded=noと同じ意味なのが-s noであることが読み取れます。最後の3行分はpaired-endリードの取り扱いに関するものです。が、-s reverse または--stranded=reverseをつけた場合にどのようにruleがreverseされるのかは私にはよくわかりません。

-s <yes/no/reverse>, **--stranded**=<yes/no/reverse>
whether the data is from a strand-specific assay (default: `yes`)

For `stranded=no`, a read is considered overlapping with a feature regardless of whether it is mapped to the same or the opposite strand as the feature. For `stranded=yes` and single-end reads, the read has to be mapped to the same strand as the feature. For paired-end reads, the first read has to be on the same strand and the second read on the opposite strand. For `stranded=reverse`, these rules are reversed.

-a <minqual>, **--a**=<minqual>
skip all reads with alignment quality lower than the given minimum value (default: 10 — Note: the default used to be 0 until version 0.5.4.)

-t <feature type>, **--type**=<feature type>
feature type (3rd column in GFF file) to be used, all features of other type are ignored (default, suitable for RNA-Seq analysis using an [Ensembl GTF file](#): `exon`)

-i <id attribute>, **--idattr**=<id attribute>
GFF attribute to be used as feature ID. Several GFF lines with the same feature ID will be considered as parts of the same feature. The feature ID is used to identity the counts in the output table. The default, suitable for RNA-Seq analysis using an [Ensembl GTF file](#), is `gene_id`.

--additional-attr=<id attributes>
Additional feature attributes, which will be printed as an additional column after the primary attribute column but before the counts column(s). The default is none, a suitable

v. release_0.10.0

デフォルトはGTF

アノテーションファイル<gff_file>の説明部分で、GFF2形式のページに飛ばされていた。このため、おそらくGTF形式のことだろうとは予想していたが、②の記述を眺めることでデフォルトはGTFだと確信する。

When <alignment_file> is paired end sorted by position, allow only so many reads to stay in memory until the mates are found (raising this number will use more memory). Has no effect for single end or paired end sorted by name. (default: 30000000)

-s <yes/no/reverse>, **--stranded**=<yes/no/reverse>
whether the data is from a strand-specific assay (default: `yes`)

For `stranded=no`, a read is considered overlapping with a feature regardless of whether it is mapped to the same or the opposite strand as the feature. For `stranded=yes` and single-end reads, the read has to be mapped to the same strand as the feature. For paired-end reads, the first read has to be on the same strand and the second read on the opposite strand. For `stranded=reverse`, these rules are reversed.

-a <minaqal>, **--a**=<minaqal>
skip all reads with alignment quality lower than the given minimum value (default: 10 — Note: the default used to be 0 until version 0.5.4.)

-t <feature type>, **--type**=<feature type>
feature type (3rd column in GFF file) to be used, all features of other type are ignored (default, suitable for RNA-Seq analysis using an [Ensembl GTF file](#): `exon`)

-i <id attribute>, **--idattr**=<id attribute>
GFF attribute to be used as feature ID. Several GFF lines with the same feature ID will be considered as parts of the same feature. The feature ID is used to identity the counts in the output table. The default, suitable for RNA-Seq analysis using an [Ensembl GTF file](#), is `gene_id`.

--additional-attr=<id attributes>
Additional feature attributes, which will be printed as an additional column after the primary attribute column but before the counts column(s). The default is none, a suitable

FAQ

①少しページ下部に移動。②FAQ。③HTSeqは、複数個所にマップされるリードは無視するポリシーなのです。

-h, --help
Show a usage summary and exit

② Frequently asked questions

My shell reports "command not found" when I try to run "htseq-count". How can I launch the script?

The file "htseq-count" has to be in the system's [search path](#). By default, Python places it in its script directory, which you have to add to your search path. A maybe easier alternative is to write `python -m HTSeq.scripts.count` instead of `htseq-count`, followed by the options and arguments, which will launch the htseq-count script as well.

Why are multi-mapping reads and reads overlapping multiple features discarded rather than counted for each feature?

The primary intended use case for `htseq-count` is *differential* expression analysis, where one compares the expression of the same gene across samples and not the expression of different genes within a sample. Now, consider two genes, which share a stretch of common sequence such that for a read mapping to this stretch, the aligner cannot decide which of the two genes the read originated from and hence reports a multiple alignment. If we discard all such reads, we undercount the total output of the genes, but the *ratio* of expression strength (the "fold change") between samples or experimental condition will still be correct, because we discard the same fraction of reads in all samples. On the other hand, if we counted these reads for both genes, a subsequent differential-expression analysis might find false positives: Even if only one of the gene changes increases its expression in reaction to treatment, the additional read caused by this would be counted for both genes, giving the wrong appearance that both genes reacted to the treatment.

v. release 0.10.0

I have used a GTF file generated by the Table Browser function of the UCSC Genome Browser, and most reads are counted as ambiguous. Why?

ambiguous

HTSeq(のデフォルトである①unionモード)では、②複数の領域にまたがってマップされるリードは、ambiguousとなる。確かにそうでしたね。

Counting reads in features ...

The following figure illustrates the effect of these three modes and the `--nonunique` option:

	union	intersection_strict	intersection_nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous (both genes with --nonunique all)	gene_A	gene_A
	ambiguous (both genes with --nonunique all)	ambiguous	ambiguous (both genes with --nonunique all)

v: release_0.10.0

FAQ

①一番下まで移動。②このあたりを眺めることで、GFF3ファイルの読み込ませ方や、`gene`や`exon`以外の任意のfeatureを用いてカウント情報を取得する場合のオプションの指定法を試行錯誤しながら学ぶ。

Counting reads in features ... x

http://htseq.readthedocs.io/en/release_0.10.0/count.html

problematic. Now, several years later, I have seen very few cases where the default `union` would not be appropriate and hence tend to recommend to just stick to `union`.

I have a GTF file? How do I convert it to GFF?

No need to do that, because GTF is a tightening of the GFF format. Hence, all GTF files are GFF files, too. By default, `htseq-count` expects a GTF file.

I have a GFF file, not a GTF file. How can I use it to count RNA-Seq reads?

The GTF format specifies, inter alia, that exons are marked by the word `exon` in the third column and that the gene ID is given in an attribute named `gene_id`, and `htseq-count` expects these words to be used by default. If your GFF file uses a word other than `exon` in its third column to mark lines describing exons, notify `htseq-count` using the `--type` option. If the name of the attribute containing the gene ID for exon lines is not `gene_id`, use the `--idattr`. Often, it is, for example, `Parent`, `GeneID` or `ID`. Make sure it is the gene ID and not the exon ID.

How can I count overlaps with features other than genes/exons?

If you have a GFF file listing your features, use it together with the `--type` and `--idattr` options. If your feature intervals need to be computed, you are probably better off writing your own counting script (provided you have some knowledge of Python). Follow the tutorial in the other pages of this documentation to see how to use HTSeq for this.

How should I cite htseq-count in a publication?

Please cite HTSeq as follows: S Anders, T P Pyl, W Huber: *HTSeq — A Python framework to work with high-throughput sequencing data*. bioRxiv 2014. doi: 10.1101/002824. (This is a preprint currently under review. We will replace this with the reference to the final published version once available.)

HTSeq 0.10.0 documentation »

v. release_0.10.0
previous | next | index

© Copyright 2010, Simon Anders. Created using Sphinx 1.7.4.

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

htseq-count実行

①乳酸菌ゲノムへのマッピング結果BAMファイル（機能ゲノム学第4回のスライド103）と、②対応するGFF3ファイルを入力として、③htseq-countでgeneレベルのカウントデータを得る例を示します。

Counting reads in features ... x

http://htseq.readthedocs.io/en/release_0.10.0/count.html

read

gene_A gene_B

alignment_not_unique
(both genes with --nonunique all)

Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

```
htseq-count [options] <alignment_files> <gff_file>
```

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```

The `<alignment_files>` are one or more files containing the aligned reads in SAM format. ([SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as [STAR](#). HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

The `<gff_file>` contains the features in the GFF format.

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

v. release_0.10.0

Linux環境でのHTSeqのインストール手順は、①乳酸菌学会誌のNGS連載第4回の、②ウェブ資料PDF中のW14-4にあります。

HTSeqのインストール

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページのR関連部分は、[インストール](#) | についての推奨手順 ([Windows2018.03.12版](#)と[Macintosh2015.04.03版](#))に従ってフリーソフトRと必要

([Windows2015.04.03](#))
(2015/04/03)

What's new?

- 「マップ後 | カウ
- 「イントロ | ファイ
- 「[H29年度NGS](#)

- 書籍 | [トランスクリプトーム解析 | 4.3.4 他の実験デザイン\(3群間\)](#) (last modified 2014/04/28)
- 書籍 | [日本乳酸菌学会誌 | について](#) (last modified 2018/05/10) **NEW**
- 書籍 | [日本乳酸菌学会誌 | 第1回イントロダクション](#) (last modified 2016/12/22)
- 書籍 | [日本乳酸菌学会誌 | 第2回GUI環境からコマンドライン環境へ](#) (last modified 2015/11/26)
- 書籍 | [日本乳酸菌学会誌 | 第3回Linux環境構築からNGSデータ取得まで](#) (last modified 2017/07/02)
- 書籍 | [日本乳酸菌学会誌 | 第4回クオリティコントロールとプログラムのインストール](#) (last modified 2018/05/10) **NEW**
- 書籍 | [日本乳酸菌学会誌 | 第5回アセンブル、マッピング、そしてQC](#) (last modified 2017/06/25)
- 書籍 | [日本乳酸菌学会誌 | 第6回ゲノムアセンブリ](#) (last modified 2017/06/21)
- 書籍 | [日本乳酸菌学会誌 | 第7回コンテナ環境構築からNGSデータ取得まで](#) (last modified 2017/06/25)

書籍 | 日本乳酸菌学会誌 | 第4回クオリティコントロールとプログラムのインストール **NEW**

日本乳酸菌学会誌の第4回分です。Linuxコマンドのリンク先は主に[日経BP社](#)様です。ウェブ資料は、共有フォルダ関連の記述(W4-5から4-8周辺)を(8/14, 8/23に引き続いて)2015年9月18日にアップデートしました。それでもまだ設定がリセットされるという方はお手数ですがお知らせ願います。

- [原稿PDF](#)
- ウェブ資料PDF(オリジナル版; 原稿PDFと同じく、HDD150GB、約1.3億の全リードファイルからスタート。)
 - [Windows用](#)(2015.09.18版; 約24MB)
 - Macintosh用(未着手)
- ウェブ資料PDF(軽量版; 原稿PDFと異なり、HDD100GB、最初の150万リードのみのファイルからスタート。)
 - [Windows用](#)(2018.05.10版; 約21MB; 推奨)
 - Macintosh用(未着手)

Linuxコマンド

- [apt-get](#) (パッケージのインストールやアップデート)
- [cd](#) (ディレクトリを変更)

htseq-count実行(したつもり)

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページのR関連部分は、[インストール](#)についての推奨手順([Windows2018.03.12版](#)と[Macintosh2015.04.03版](#))に従ってフリーソフトRと必要なパッケージをインストール済みであるという前提で記述しています。初心者の方には[基本的な利用法](#)

([Windows2015](#) | [Macintosh2015](#))

- [マップ後 | 出力ファイルの読み込み](#) | [htSeqTools\(Planet 2012\)](#) (last modified 2013/06/19)
- [マップ後 | カウント情報取得](#) | (last modified 2018/05/30) **NEW**
- [マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有](#) | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/29) **NEW**
- [マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有](#) | [HTSeq\(Anders 2015\)](#) (last modified 2018/05/30) **NEW**
- [マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション無](#) | [QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/26) **NEW**
- [マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション有](#) | [QuasR\(Gaidatzis 2015\)](#) (last modified 2016/02/13)
- [マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション無](#) | [QuasR\(Gaidatzis 2015\)](#) (last modified 2015/07/02)

What's new?

- 「マップ後 | カ
- 「イントロ | フ
- 「H29年度N

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4_3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | [ファイル形式の変換 | GFF3 → GTF](#)」の例題1で作成したものです。また、[sample RNAseq4_3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

htseq-count実行コマンド

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GFF3 gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GFF3 transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GFF3 exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

htseq-count実行

①乳酸菌ゲノムへのマッピング結果BAMファイル(hoge.bam)と②対応するGFF3ファイル(hoge.gff3)が、③htseq-countを実行する上で絶対必要な入力ファイルに相当するものです。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

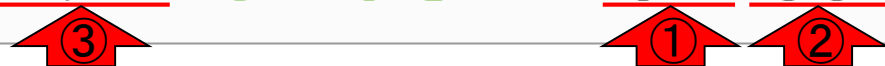
HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis_2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。

対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge1.gtf)は、「イントロ | ファイル形式の変換 | GFF3 -> GTF」の例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので、hoge.bamとして取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定(gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```



2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定(transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

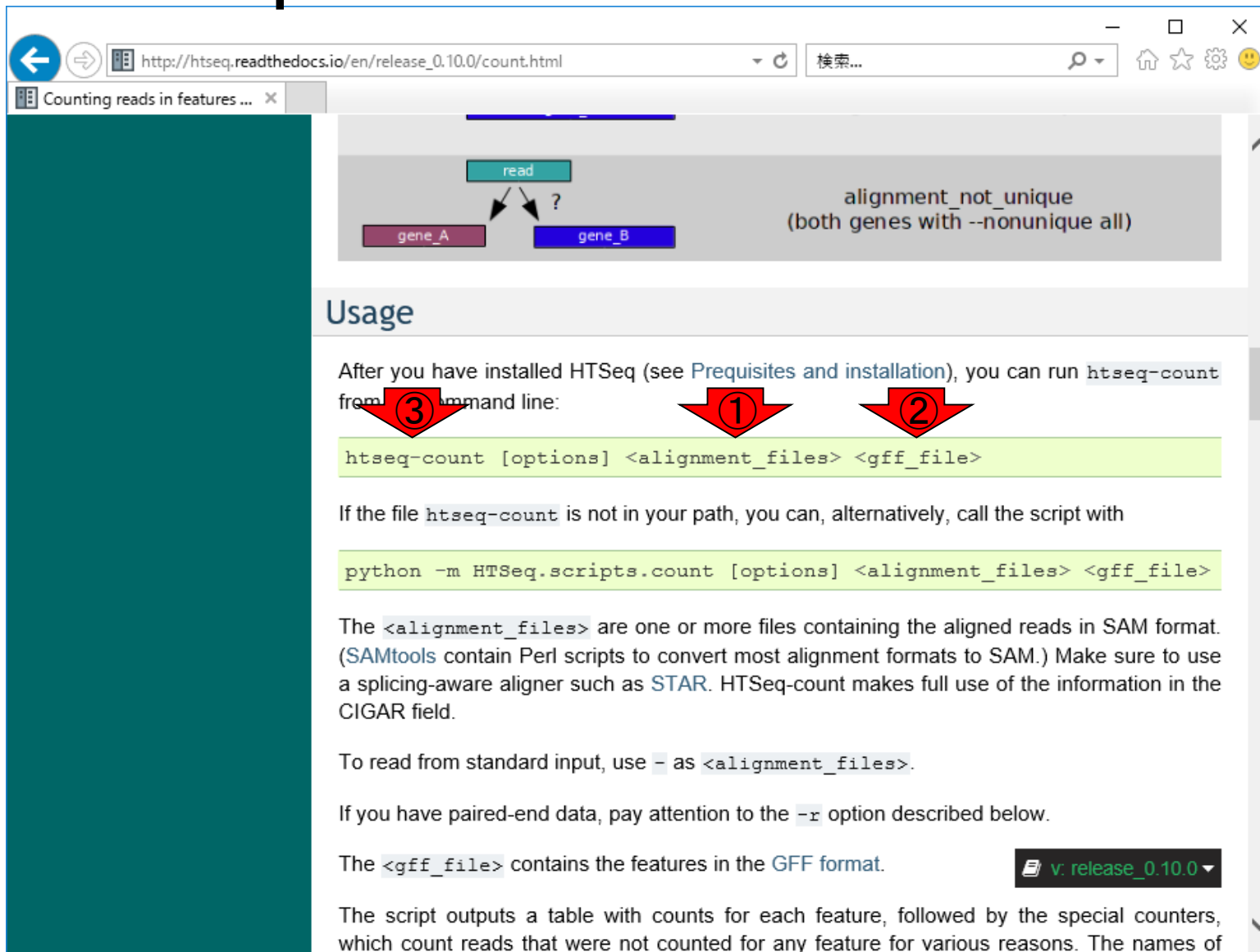
```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定(exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```


htseq-count実行コマンド



Counting reads in features ... x

read

gene_A gene_B

alignment_not_unique
(both genes with --nonunique all)

Usage

After you have installed HTSeq (see [Prerequisites and installation](#)), you can run `htseq-count` from the command line:

③ `htseq-count` **①** `[options]` **②** `<alignment_files>` `<gff_file>`

If the file `htseq-count` is not in your path, you can, alternatively, call the script with

```
python -m HTSeq.scripts.count [options] <alignment_files> <gff_file>
```

The `<alignment_files>` are one or more files containing the aligned reads in SAM format. ([SAMtools](#) contain Perl scripts to convert most alignment formats to SAM.) Make sure to use a splicing-aware aligner such as [STAR](#). HTSeq-count makes full use of the information in the CIGAR field.

To read from standard input, use `-` as `<alignment_files>`.

If you have paired-end data, pay attention to the `-r` option described below.

The `<gff_file>` contains the features in the GFF format. v. release_0.10.0

The script outputs a table with counts for each feature, followed by the special counters, which count reads that were not counted for any feature for various reasons. The names of

hoge.gff3

マップした乳酸菌ゲノムに対応するGFF3ファイルの大元は①ですが、赤下線部分でも書いてあるように、見やすくする目的で②hoge.gff3と短いファイル名にしてあります。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。



1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```



2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

hoge.bam

①乳酸菌ゲノムへのマッピング結果BAMファイル(hoge.bam)についても同様で、大元は②です。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47のFASTA形式ファイル\(sample RNAseq4.fa\)](#)です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12AのFASTA形式ゲノム配列ファイル\(Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa\)](#)です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定(gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

①

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定(transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定(exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

オプション[options]

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```



2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

オプション -f

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | H

①マッピング結果がBAMファイルの場合は、②-f bamとしなければならない。SAMファイルの場合は、②はなくてもよいが-f samと書いてもよい。

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```



2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```


オプション-tと-i

マップ後 | カウント情報取得 | single-end |

今やろうとしているのは、GFF3ファイルを入力とした、①geneレベルのカウントデータ取得。但し、デフォルトのアノテーションファイルはGTFなので、②GFF3ファイルを読み込ませる場合は、3列目が③geneで9列目の④gene_idから始まる箇所の情報(feature)を取り扱ふと明示してやらねばならない。この、「3列目が③geneで9列目の④gene_id」に相当するのが…

HTSeqというPythonプログラムを用いてカウント情報を得るやり方

アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)の例題10を実行して得られたものは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする

([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ」ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4_3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。294 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

hoge.gff3

①3列目のgeneと、②9列目の③gene_idです。geneレベルのカウントデータがほしいのだから、①のみの指定でいいじゃないか(その指定だけでどうにかして!)、という気持ちもわからないではない。しかしながら、9列目には多数の情報が記載されているため、9列目内のどこのfeature(情報、という理解でよい)を取り扱うかを明示しなければ困るからだと思えばよい。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	##gff-version	3												
2	##sequence-region	Chromosome	360	2277853										
3	#!genome-build	European Nucleotide Archive	ASM82939v1											
4	#!genome-version	GCA_000829395.1												
5	#!genome-date	2014-11												
6	#!genome-build-accession	GCA_000829395.1												
7	#!genebuild-last-updated	2014-11												
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=pr					
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_10					
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;cc					
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;pr					
12	###													
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=pr					
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_10					
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;cc					
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;pr					

例題1実行(したつもり)

①例題1のコマンドを実行したつもりで、②出力ファイル(output_GFF3_gene.txt)を眺める。③にあります。④2,194 genesからなります。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis_2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。

対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge1.gtf)は、「イントロ | ファイル形式の変換 | GFF3 -> GTF」の例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので、hoge.bamとして取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定(gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定(transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定(exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

output_GFF3_gene.txt

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(A

①例題1の③実行結果ファイル(output_GFF3_gene.txt)の、
④最初の7行と⑤最後の7行。
⑥が2,194行目に相当します。

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後|カウント情報取得|single-end|ゲノム|アノテーション有|QuasR(Gaidatzis 2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)は、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aのゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge.gtf)とGTF形式の変換|GFF3->GTFの例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature指定(gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合sam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature指定(transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合sam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature指定(exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合sam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0

LOOC260_122680	0
LOOC260_122690	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

output_GFF3_gene.txt

①1列目が指定したfeatureで、②2列目がカウント数。③この結果ファイルには、最後の5行分にログ情報が含まれている。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。は、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aのゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge.gtf)とGFF3形式の変換[GFF3 -> GTF]の例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので取り扱います。

LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature指定 (gene_idの代わりにIDでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature指定 (transcript_idの代わりにIDやParentでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

LOOC260_122680	0
LOOC260_122690	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature指定 (exon_idの代わりにParentやNameでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

output_GFF3_gene.txt

①②③最初の3 features上に、計10リード分カウントされている。また、④1リードがambiguous扱いになっていることもわかる。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12.1ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)で、対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosomes.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイルの変換 | GFF3 -> GTFの例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので、hoge.bamとして取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature指定 (gene_idの代わりにIDでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合にはsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature指定 (transcript_idの代わりにIDやParentでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合にはsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature指定 (exon_idの代わりにParentやNameでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合にはsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0

LOOC260_122680	0
LOOC260_122690	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

おさらい

②

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939v
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
Chromosome ena gene 360 1676 . + ③ ID=gene:
Chromosome ena transcript 360 1676 . + . ID=trans
Chromosome ena exon 360 1676 . + . Parent=t
Chromosome ena CDS 360 1676 . + 0 ID=CDS:
###
Chromosome ena gene 1852 2991 . + ③ ID=gene:
Chromosome ena transcript 1852 2991 . + . ID=trans
Chromosome ena exon 1852 2991 . + . Parent=t
Chromosome ena CDS 1852 2991 . + 0 ID=CDS:
###
Chromosome ena gene 3233 3457 . + ③ ID=gene:
Chromosome ena transcript 3233 3457 . + . ID=trans
Chromosome ena exon 3233 3457 . + . Parent=t
Chromosome ena CDS 3233 3457 . + 0 ID=CDS:
###
Chromosome ena gene 3467 4588 . + ③ ID=gene:
```

①マップする側 (sample_RNAseq4.fa) の計 11リードは、②hoge.gff3中の③gene領域を参考にしながら作成。全て完全一致でマップされるように設計。

①

```
>Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
>Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
>Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
>Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAGAAC
```

対応関係のおさらい

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939v
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
```


Chromosome	ena	gene	360	1676	.	+	①	ID=gene:
Chromosome	ena	transcript	360	1676	.	+	.	ID=trans
Chromosome	ena	exon	360	1676	.	+	.	Parent=t
Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:
###								
Chromosome	ena	gene	1852	2991	.	+	②	ID=gene:
Chromosome	ena	transcript	1852	2991	.	+	.	ID=trans
Chromosome	ena	exon	1852	2991	.	+	.	Parent=t
Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:
###								
Chromosome	ena	gene	3233	3457	.	+	③	ID=gene:
Chromosome	ena	transcript	3233	3457	.	+	.	ID=trans
Chromosome	ena	exon	3233	3457	.	+	.	Parent=t
Chromosome	ena	CDS	3233	3457	.	+	0	ID=CDS:
###								
Chromosome	ena	gene	3467	4588	.	+	.	ID=gene:

①2リードが領域[360, 1676]、②5リードが領域[1852, 2991]、③3リードが領域[3233, 3457]の少なくとも一部に被るように設計されていた。また、④最後の1リードが複数の遺伝子領域にまたがるように設計されている。


```
① { >Chromosome_361_400
TGACTGATTTAGAAACACTTTGGGACACAATTAAGAATC
>Chromosome_1637_1676
AGAAGATGTCCAAAACCTTAAAATGGAGCTAAAGCCATAG
② { >Chromosome_1851_1890
CATGAAATTTACAATTAGTCGTGCAACTTTTACAGCCAAA
>Chromosome_1843_1882
TAACCAATCATGAAATTTACAATTAGTCGTGCAACTTTTA
>Chromosome_1833_1872
CTTCAAGGAGTAACCAATCATGAAATTTACAATTAGTCGT
>Chromosome_1823_1862
CAAATTCAACCTTCAAGGAGTAACCAATCATGAAATTTAC
③ { >Chromosome_1813_1852
AAATTAAGACAAATTCAACCTTCAAGGAGTAACCAATCA
>Chromosome_3418_3457
GATTGCAGATAATGGGACATTTGTCATTCAAATGAGTAG
>Chromosome_3420_3459
TTGCAGATAATGGGACATTTGTCATTCAAATGAGTAGGC
④ { >Chromosome_3422_3461
GCAGATAATGGGACATTTGTCATTCAAATGAGTAGGCAA
>Chromosome_3443_3482
ATTCAAATGAGTAGGCAACTTAAATGATTTTAAAAGAAC
```

それゆえ...

##gff-version 3							
##sequence-region Chromosome 360 2277853							
#!genome-build European Nucleotide Archive ASM82939v							
#!genome-version GCA_000829395.1							
#!genome-date 2014-11							
#!genome-build-accession GCA_000829395.1							
#!genebuild-last-updated 2014-11							
Chromosome	ena	gene	360	1676	.	+	ID=gene:
Chromosome	ena	transcript	360	1676	.	+	ID=trans:
Chromosome	ena	exon	360	1676	.	+	Parent=t
Chromosome	ena	CDS	360	1676	.	+ 0	ID=CDS:
###							
Chromosome	ena	gene	1852	2991	.	+	ID=gene:
Chromosome	ena	transcript	1852	2991	.	+	ID=trans:
Chromosome	ena	exon	1852	2991	.	+	Parent=t
Chromosome	ena	CDS	1852	2991	.	+ 0	ID=CDS:
###							
Chromosome	ena	gene	3233	3457	.	+	ID=gene:
Chromosome	ena	transcript	3233	3457	.	+	ID=trans:
Chromosome	ena	exon	3233	3457	.	+	Parent=t
Chromosome	ena	CDS	3233	3457	.	+ 0	ID=CDS:
###							
Chromosome	ena	gene	3467	4588	.	+	ID=gene:



LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0



LOOC260_122680	0
LOOC260_122690	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

featureは...

①がfeatureであり、②赤下線部分の情報を抽出して得られていることが分かります。

	A	B	C	D	E	F	G	H	I	J	L
1	##gff-version	3									
2	##sequence-region	Chromosome	360	2277853							
3	#!genome-build	European Nucleotide Archive	ASM82939v1								
4	#!genome-version	GCA_000829395.1									
5	#!genome-date	2014-11									
6	#!genome-build-accession	GCA_000829395.1									
7	#!genebuild-last-updated	2014-11									
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein	LOOC260_100010	2
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=	LOOC260_100020	5
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Na	LOOC260_100030	3
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=trans	LOOC260_100040	0
12	###									LOOC260_100050	0
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name	LOOC260_100060	0
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=	LOOC260_100070	0
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Na	__no_feature	0
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=trans	__ambiguous	1
										__too_low_aQual	0
										__not_aligned	0
										__alignment_not_unique	0

指定したオプション

①の情報を抽出すべく、htseq-count実行時に指定したオプションが、(3列目の)②-t geneと、(9列目の)③-i gene_idだったことを思い出そう。

	A	B	C	D	E	F	G	H	I	J	K	L	
1	##gff-version	3											
2	##sequence-region	Chromosome	360	2277853									
3	#!genome-build	European Nucleotide Archive	ASM82939v1										
4	#!genome-version	GCA_000829395.1											
5	#!genome-date	2014-11											
6	#!genome-build-accession	GCA_000829395.1											
7	#!genebuild-last-updated	2014-11											
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=replication initiation protein DnaA;gene_id=LOOC260_100010;logic_name=LOOC260_100010			2	
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=transcript:LOOC260_100010			5	
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=transcript:LOOC260_100010			3	
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:LOOC260_100010			0	
12	###											0	
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaA;biotype=protein_coding;description=replication initiation protein DnaA;gene_id=LOOC260_100020;logic_name=LOOC260_100020			0	
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=transcript:LOOC260_100020			1	
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=transcript:LOOC260_100020			0	
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:LOOC260_100020			0	
												__no_feature	0
												__ambiguous	1
												__too_low_aQual	0
												__not_aligned	0
												__alignment_not_unique	0

区切り文字;で分割

単に③gene_idから始まる文字列だと、③gene_id以降の文字列を全部抽出してしまうことになる。が、例えば③gene_id=から④区切り文字;までの文字列を抽出すれば①の部分のみを抽出できる。

自動保存 日 ↶ ↷ hoge.gff3

ファイル ホーム 挿入 ページレイアウト 数式 テータ 校閲 表示 実行したい作業を入力してください 共有

18 : X ✓ fx ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;descr LOOC260_100010 2
 replication initiation protein DnaA;gene_id=LOOC260_100010;logic_na LOOC260_100020 5

	A	B	C	D	E	F	G	H	I	J	K	
1	##gff-version	3										LOOC260_100030 3
2	##sequence-region	Chromosome 360	2277853									LOOC260_100040 0
3	#!genome-build	European Nucleotide Archive	ASM82939v1									LOOC260_100050 0
4	#!genome-version	GCA_000829395.1										LOOC260_100060 0
5	#!genome-date	2014-11										LOOC260_100070 0
6	#!genome-build-accession	GCA_000829395.1										
7	#!genebuild-last-updated	2014-11										
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;descr			
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=transcript:LOOC260_100010			
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=transcript:LOOC260_100010			LOOC260_122680 0
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:LOOC260_100010			LOOC260_122690 0
12	###											__no_feature 0
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaA;biotype=protein_coding;descr			__ambiguous 1
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=transcript:LOOC260_100020			__too_low_aQual 0
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=transcript:LOOC260_100020			__not_aligned 0
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:LOOC260_100020			__alignment_not_unique 0

hoge

準備完了

区切り文字;で分割

このあたりは、例えばawkコマンド(スライド22あたり)を用いて、区切り文字;で文字列を分割する戦略を経験上知っていれば、おそらく内部的にそのようなことをやっているのだろうと想像がつく。

	A	B	C	D	E	F	G	H	I	J	K		
1	##gff-version	3										LOOC260_100010	2
2	##sequence-region	Chromosome	360	2277853								LOOC260_100020	5
3	#!genome-build	European Nucleotide Archive	ASM82939v1									LOOC260_100030	3
4	#!genome-version	GCA_000829395.1										LOOC260_100040	0
5	#!genome-date	2014-11										LOOC260_100050	0
6	#!genome-build-accession	GCA_000829395.1										LOOC260_100060	0
7	#!genebuild-last-updated	2014-11										LOOC260_100070	0
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein				
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=				
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=			LOOC260_122680	0
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=trans			LOOC260_122690	0
12	###											__no_feature	0
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=			__ambiguous	1
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=			__too_low_aQual	0
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=			__not_aligned	0
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=trans			__alignment_not_unique	0

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

exonレベルのカウントデータ

exonレベルのカウントデータを取得したい場合は、①exonと②exon_idをオプションとして与えればよいだろうと想像し、実践する。

自動保存 hoge.gff3 保存しました

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 実行したい作業を入力してください 共有

I10 : Parent=transcript:BAP84581;Name=BAP84581-1;constitutive=1;ensembl_end_phase=0;ensembl_phase=0;exon_id=BAP84581-1;rank=1;version=1

	A	B	C	D	E	F	G	H	I	J	K	M	N	O
1	##gff-version 3													
2	##sequence-region			Chromosome	360	2277853								
3	#!genome-build			European Nucleotide Archive	ASM82939v1									
4	#!genome-version			GCA_000829395.1										
5	#!genome-date			2014-11										
6	#!genome-build-accession			GCA_000829395.1										
7	#!genebuild-last-updated			2014-11										
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein					
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010					
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constitutive=1					
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_coding=1					
12	###													
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein					
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020					
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constitutive=1					
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein_coding=1					

hoge 100%

準備完了

exonレベルのカウント

①例題3が、exonレベルのカウントデータを取得するやり方。②exonと③exon_idをオプションとして与えて実行した④2,262 exonsからなる⑤出力ファイル(output_GFF3_exon.txt)が…

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定(gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定(transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定(exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

output_GFF3_exon.txt

これ (output_GFF3_exon.txt)。①最初の7行と、②最後の7行。③が2,262行目に相当します。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis_2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されている Lactobacillus casei 12Aのゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.dna.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge.gtf)は、GTF形式の変換 | GFF3 -> GTFの例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature指定 (gene_idの代わりにIDでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature指定 (transcript_idの代わりにIDやParentでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature指定 (exon_idの代わりにParentやNameでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

BAP84581-1	2
BAP84582-1	5
BAP84583-1	3
BAP84584-1	0
BAP84585-1	0
BAP84586-1	0
BAP84587-1	0

LOOC260_116770-1	0
LOOC260_121590-1	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イントロ、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

課題1

htseq-countプログラムを用いて、transcriptレベルのカウントデータを得たい場合に入力するオプションを示すべく、①下線部分の空欄を埋めよ。但し、マッピング結果はBAMファイル、アノテーションファイルはGFF3形式、出力ファイル名はu.txtとする。

自動保存 日 ↶ ↷ = hoge.gff3

ファイル ホーム 挿入 ページレイアウト 数式 テータ 校閲 表示

ID=transcript:BAP84581;Parent=gene:LOOC260_100010;Name=dnaA-1;biotype=protein_coding;transcript_id=BAP84581;version=1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version	3													
2	##sequence-region	Chromosome	360	2277853											
3	#!genome-build	European Nucleotide Archive	ACM82020v1												
4	#!genome-build	European Nucleotide Archive	ACM82020v1												
5	#!genome-date	2014-11													
6	#!genome-build-accession	GCA_000829395.1													
7	#!genebuild-last-updated	2014-11													
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding						
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010						
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constitutive_exon						
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_coding						
12	###														
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding						
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020						
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constitutive_exon						
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein_coding						

```
htseq-count -t _____ -i _____ -f bam hoge.bam hoge.gff3 > u.txt
```

準備完了

課題2

htseq-countプログラムを用いて、CDSレベルのカウントデータを得たい場合に入力するオプションを示すべく、①下線部分の空欄を埋めよ。但し、マッピング結果はSAMファイル(hoge.sam)、アノテーションファイルはGFF3形式、出力ファイル名はk.txtとする。

自動保存 hoge.gff3

ファイル ホーム 挿入 ページレイアウト 数式 テータ 校閲 表示

l11 : X ✓ fx ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_id=BAP84581

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version 3														
2	##sequence-region		Chromosome	360	2277853										
3	#!genome-build		European Nucleotide Archive	ACM82020v1											
4	#!genome-date		2014-11												
5	#!genome-build-accession		GCA_000829395.1												
6	#!genebuild-last-updated		2014-11												
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein						
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010						
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constit						
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein						
12	###														
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein						
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020						
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constit						
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein						

htseq-count -t _____ -i _____ -f _____ hoge.sam hoge.gff3 > k.txt

準備完了

課題3

課題2で得られる出力ファイル①k.txtにおいて、1列目のfeature IDはマッピング結果ファイルの種類(SAMまたはBAM)とは②k.txtの最初の2行分のfeature IDを示すべく、下線部分の空欄を埋めよ。

```
htseq-count -t _____ -i _____ -f _____ hoge.sam hoge.gff3 > k.txt
```

①

② k.txt の場合 : +

1 行目が _____

2 行目が _____

課題1~3についてはhoge.gff3
ファイルを見ればわかります。

課題1~3のヒント

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version	3													
2	##sequence-region	Chromosome	360	2277853											
3	#!genome-build	European Nucleotide Archive	ASM82939v1												
4	#!genome-version	GCA_000829395.1													
5	#!genome-date	2014-11													
6	#!genome-build-accession	GCA_000829395.1													
7	#!genebuild-last-updated	2014-11													
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein						
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010						
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constitutive						
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_id=BAP84581						
12	###														
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein						
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020						
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constitutive						
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein_id=BAP84582						

課題1~3のヒント

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **①** **W**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GFF3 gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GFF3 transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GFF3 exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

htseq-countページのFAQ

応用スキルの習得や課題関連の事柄については、①htseq-countの、②一番下の、③のあたりにも情報あり。

The screenshot shows the documentation page for htseq-count. A red box highlights the FAQ section titled "I have a GFF file, not a GTF file. How can I use it to count RNA-Seq reads?". A red arrow labeled "3" points to the left margin. Another red arrow labeled "2" points to the bottom navigation bar. The page content includes:

I have a GTF file? How do I convert it to GFF?
No need to do that, because GTF is a tightening of the GFF format. Hence, all are GFF files, too. By default, htseq-count expects a GTF file.

I have a GFF file, not a GTF file. How can I use it to count RNA-Seq reads?
The GTF format specifies, inter alia, that exons are marked by the word `exon` in the third column and that the gene ID is given in an attribute named `gene_id`, and htseq-count expects these words to be used by default. If your GFF file uses a word other than `exon` in its third column to mark lines describing exons, notify htseq-count using the `--type` option. If the name of the attribute containing the gene ID for exon lines is not `gene_id`, use the `--idattr`. Often, it is, for example, `Parent`, `GeneID` or `ID`. Make sure it is the gene ID and not the exon ID.

How can I count overlaps with features other than genes/exons?
If you have a GFF file listing your features, use it together with the `--type` and `--idattr` options. If your feature intervals need to be computed, you are probably better off writing your own counting script (provided you have some knowledge of Python). Follow the tutorial in the other pages of this documentation to see how to use HTSeq for this.

How should I cite htseq-count in a publication?
Please cite HTSeq as follows: S Anders, T P Pyl, W Huber: *HTSeq — A Python framework to work with high-throughput sequencing data*. bioRxiv 2014. doi: 10.1101/002824. (This is a preprint currently under review. We will replace this with the reference to the final published version once available.)

HTSeq 0.10.0 documentation »

v. release_0.10.0
previous | next | index

© Copyright 2010, Simon Anders. Created using Sphinx 1.7.4.

1. 平成30年06月12日 (PC使用)
講義資料PDF
(Rで)塩基配列解析
QuasR : Gaidatzis et al., Bioinformatics, 2015
HTSeq : Anders et al., Bioinformatics, 2015
hoge10.txt
htseq-countのページ



Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

課題4のイントロ

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

①

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

課題4のイントロ

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4_3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ」ファイル形式の変換 | [GFF3 -> GTF](#) | 参照してください。取り扱います。

1. GFF3でgeneレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (gene_idの代わりにIDでもOK) してファイルは [output_GFF3_gene.txt](#) で

```
htseq-count -t gene -i gene
```

2. GFF3でtranscriptレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (transcript_idの代わりにIDやParent、Name、場合sam)。出力ファイルは [output](#)

```
htseq-count -t transcript
```

3. GFF3でexonレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (exon_idの代わりにParentやName、場合sam)。出力ファイルは [output_GFF3](#)

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 4:08午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls [ 4:08午後 ]
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i gene_id -f bam hoge.b [ 4:09午後 ]
am hoge.gff3 > output_GFF3_gene.txt
8980 GFF lines processed.
11 SAM alignments processed.
iu@bielinux[mac_share] █
```

課題4のイントロ

①が実行コマンド。②のような実行ログが表示されて、エラーを吐くことなく無事終了します。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4_3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample RNAseq4.fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ」ファイル形式の変換 | [GFF3 -> GTF](#) | 参照してください。取り扱います。

1. GFF3でgeneレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (gene_idの代わりにIDでもOK) してファイルは [output_GFF3_gene.txt](#) で

```
htseq-count -t gene -i gene
```

2. GFF3でtranscriptレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (transcript_idの代わりにIDやParent、Name、場合sam)。出力ファイルは [output](#)

```
htseq-count -t transcript
```

3. GFF3でexonレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (exon_idの代わりにParentやName、場合sam)。出力ファイルは [output_GFF3_exon.txt](#)

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i gene_id -f bam hoge.b
am hoge.gff3 > output_GFF3_gene.txt
8980 GFF lines processed.
11 SAM alignments processed.
iu@bielinux[mac_share]
```



課題4のイントロ

①出力ファイル(output_GFF3_gene.txt)の最初の5行分を表示。妥当な結果ですね。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis_2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。

対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge1.gtf)は、「イントロ」ファイル形式の変換 | GFF3 -> GTF | 参照してください。取り扱います。

1. GFF3でgeneレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (gene_idの代わりにIDでもOK) してファイルはoutput_GFF3_gene.txtで

```
htseq-count -t gene -i gene
```

2. GFF3でtranscriptレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (transcript_idの代わりにIDやParentの場合はsam)。出力ファイルはoutput_GFF3_transcript.txt

```
htseq-count -t transcript
```

3. GFF3でexonレベルのカウントデータ取得

アノテーションファイルがGFF3形式 (exon_idの代わりにParentやNameの場合はsam)。出力ファイルはoutput_GFF3_exon.txt

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 4:20午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls [ 4:20午後 ]
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i gene_id -f bam hoge.b [ 4:20午後 ]
am hoge.gff3 > output_GFF3_gene.txt
8980 GFF lines processed.
11 SAM alignments processed.
iu@bielinux[mac_share] head -n 5 output_GFF3_gene.txt [ 4:20午後 ]
L00C260_100010 2
L00C260_100020 5
L00C260_100030 3
L00C260_100040 0
L00C260_100050 0
iu@bielinux[mac_share] [ 4:21午後 ]
```


①でも書いているように、②の部分を-i IDとしてもうまく動きます。

-i IDでもOK

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **NEW**

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。

対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge1.gtf)は、「イントロ | ファイル形式の変換 | GFF3 -> GTF」の例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので、hoge.bamとして取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```



2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定 (transcript_idの代わりにIDやParentでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定 (exon_idの代わりにParentやNameでもOK) しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

①でも書いているように、②の部分を-i IDとしてもうまく動きます。

-i IDでもOK

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015)」の例題10を実行して得られたマッピング結果(sample RNAseq4 3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されている Lactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。

対応するGFF3形式のアノテーションファイルはLactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge1.gtf)は、「イントロ | ファイル形式の変換 | GFF3 -> GTF」で取り扱います。

1. GFF3でgeneレベルのカウントデータ

アノテーションファイルがGFF3形式 (gene_idの代わりにIDでもOK) してファイルはoutput_GFF3_gene.txtで

```
htseq-count -t gene -i gene
```

2. GFF3でtranscriptレベルのカウント

アノテーションファイルがGFF3形式 (transcript_idの代わりにIDやParent、Name、sam)。出力ファイルはoutput

```
htseq-count -t transcript
```

3. GFF3でexonレベルのカウントデータ

アノテーションファイルがGFF3形式 (exon_idの代わりにParentやName、sam)。出力ファイルはoutput_GFF3

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 4:24午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls [ 4:24午後 ]
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i ID -f bam hoge.bam ho [ 4:24午後 ]
ge.gff3 > output_GFF3_gene.txt
8980 GFF lines processed.
11 SAM alignments processed.
iu@bielinux[mac_share] █
```

-i IDでもOK

①でも書いているように、②の部分
を-i IDとしてもうまく動きます。
③妥当な結果ですね。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis_2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。

対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge1.gtf)は、「イントロ | ファイル形式の変換 | GFF3 -> GTF」で取り扱います。

1. GFF3でgeneレベルのカウントデータ取得

アノテーションファイルがGFF3形式(gene_idの代わりにIDでもOK)してファイルはoutput_GFF3_gene.txtで

```
htseq-count -t gene -i gene
```

2. GFF3でtranscriptレベルのカウントデータ取得

アノテーションファイルがGFF3形式(transcript_idの代わりにIDやParentの場合はsam)。出力ファイルはoutput_GFF3_transcript.txt

```
htseq-count -t transcript
```

3. GFF3でexonレベルのカウントデータ取得

アノテーションファイルがGFF3形式(exon_idの代わりにParentやNameの場合はsam)。出力ファイルはoutput_GFF3_exon.txt

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 4:24午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls [ 4:24午後 ]
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i ID -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt [ 4:24午後 ]
8980 GFF lines processed.
11 SAM alignments processed.
iu@bielinux[mac_share] head -n 5 output_GFF3_gene.txt [ 4:24午後 ]
gene:L00C260_100010 2
gene:L00C260_100020 5
gene:L00C260_100030 3
gene:L00C260_100040 0
gene:L00C260_100050 0
iu@bielinux[mac_share] [ 4:25午後 ]
```


-i IDでもOK

①gene_idを指定した場合は②の情報を抽出しているのに対し、③IDの場合は④の部分抽出しているから。

自動保存 hoge.gff3.xlsx - 保存しました サインイン

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 実行したい作業を入力してください 共有

18 ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal
Location is on protein DnaA;gene_id=LOOC260_100010;logic_name=ena;version=1

	A	B	C	D	E	F	G	H	I	J	L	M	N	O
1	##gff-version	3												
2	##sequence-region	Chromosome	360	2277853										
3	#!genome-build	European Nucleotide Archive	ASM82939v1											
4	#!genome-version	GCA_000829395.1												
5	#!genome-date	2014-11												
6	#!genome-build-accession	GCA_000829395.1												
7	#!genebuild-last-updated	2014-11												
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal;Location is on protein DnaA;gene_id=LOOC260_100010;logic_name=ena;version=1					
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010					
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constitutive					
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_coding					
12	###													
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding;description=chromosomal;Location is on protein DnaN;gene_id=LOOC260_100020;logic_name=ena;version=1					
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020					
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constitutive					
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein_coding					

hoge

準備完了 100%

gene symbolsで欲しい!

①のようなgene symbolsでfeature IDを取り扱えれば、特にgene symbolsをベースとする機能解析(GO解析やPathway解析)時に便利。しかし、②-i Nameとしてhtseq-countを実行してもうまくいきません。

The screenshot shows a spreadsheet with a formula bar at the top. The formula bar contains the text: `ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein`. Two red arrows point to the `Name=dnaA` and `biotype=protein_coding` parts of the string. The spreadsheet data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version	3													
2	##sequence-region	Chromosome	360	2277853											
3	#!genome-build	European Nucleotide Archive	ASM82939v1												
4	#!genome-version	GCA_000829395.1													
5	#!genome-date	2014-11													
6	#!genome-build-accession	GCA_000829395.1													
7	#!genebuild-last-updated	2014-11													
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein						
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010						
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constit						
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein						
12	###														
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein						
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020						
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constit						
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein						

-i Nameはだめ...

①のようなgene symbolsでfeature IDを取り扱えれば、特にgene symbolsをベースとする機能解析(GO解析やPathway解析)時に便利。しかし、②-i Nameとして③htseq-countを実行してもうまくいきません。

The image shows a spreadsheet and a terminal window. The spreadsheet displays a table with columns A through O and rows 1 through 16. Row 18 contains a GFF3 entry: `ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein...`. Red arrows point to the `Name=dnaA` attribute in the spreadsheet and the `-i Name` option in the terminal command. The terminal window shows the execution of `htseq-count` on a BAM file, which fails with an error: `Error occurred when processing GFF file (line 18 of file hoge.gff3): Feature gene:LOOC260_100030 does not contain a 'Name' attribute [Exception type: ValueError, raised in count.py:53]`. A red arrow points to the error message in the terminal.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version	3													
2	##sequence-region														
3	#!genome-build	Europ													
4	#!genome-version	GO													
5	#!genome-date	2014-													
6	#!genome-build-acce														
7	#!genebuild-last-upda														
8	Chromosome	ena	ger												
9	Chromosome	ena	tra												
10	Chromosome	ena	exc												
11	Chromosome	ena	CD												
12	###														
13	Chromosome	ena	ger												
14	Chromosome	ena	tra												
15	Chromosome	ena	exc												
16	Chromosome	ena	CD												

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i Name -f bam hoge.bam
hoge.gff3 > output_GFF3_gene.txt
Error occurred when processing GFF file (line 18 of file hoge.gff3):
Feature gene:LOOC260_100030 does not contain a 'Name' attribute
[Exception type: ValueError, raised in count.py:53]
iu@bielinux[mac_share]
```

エラーメッセージ

④がエラーメッセージです。この中にエラーの原因が書かれています。あえて説明はしません。

The image shows a spreadsheet application window with a GFF3 file open. The spreadsheet has columns A through O and rows 1 through 16. The formula bar shows the following text: `ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein`. Red arrows labeled ① and ② point to the `Name=dnaA` and `biotype=protein_coding` parts of the text, respectively.

Overlaid on the spreadsheet is a terminal window showing the following commands and output:

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 4:25午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls [ 4:25午後 ]
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i Name -f bam hoge.bam
hoge.gff3 > output_GFF3_gene.txt
Error occured when processing GFF file (line 18 of file hoge.gff3):
Feature gene:LOOC260_100030 does not contain a 'Name' attribute
[Exception type: ValueError, raised in count.py:53]
iu@bielinux[mac_share] [ 4:26午後 ]
```

Red arrows labeled ③ and ④ point to the terminal window. Arrow ③ points to the `htseq-count` command, and arrow ④ points to the error message box.

エラーの原因を探る

①hoge.gff3の、②18行目の記述内容を、③表示。ここを示す理由、わかりますよね。

Excel spreadsheet showing genomic data from a GFF3 file. The file name is 'hoge.gff3'. The spreadsheet displays columns for chromosome, feature type, coordinates, and ID. Row 18 is highlighted with a green box, and its content is shown in the formula bar, which is also highlighted with a red box.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
7	#!genebuild-last-updated 2014-11														
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein						
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010						
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constit						
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein						
12	###														
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein						
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020						
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constit						
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein						
17	###														
18	Chromosome	ena	gene	3233	3457	.	+	.	ID=gene:LOOC260_100030;biotype=protein_coding;desc						
19	Chromosome	ena	transcript	3233	3457	.	+	.	ID=transcript:BAP84583;Parent=gene:LOOC260_100030						
20	Chromosome	ena	exon	3233	3457	.	+	.	Parent=transcript:BAP84583;Name=BAP84583-1;constit						
21	Chromosome	ena	CDS	3233	3457	.	+	0	ID=CDS:BAP84583;Parent=transcript:BAP84583;protein						
22	###														

課題4: 原因の把握と対策

①-i Nameとして②htseq-countを実行してもうまくいかない理由について述べよ。また、①のコマンドでうまく実行できるようにすればどうすればいいか、自由に考え(戦略)を述べよ。

The image shows a terminal window on a Linux system. The terminal prompt is `iu@bielinux[~/Desktop/mac_share]`. The user has run `pwd`, `ls`, and `htseq-count -t gene -i Name -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt`. The `htseq-count` command has failed with an error: `Error occurred when processing GFF file (line 18 of file hoge.gff3): Feature gene:LOOC260_100030 does not contain a 'Name' attribute [Exception type: ValueError, raised in count.py:53]`. A red arrow labeled '1' points to the `-i Name` option in the command, and another red arrow labeled '2' points to the error message. In the background, a spreadsheet is visible with a cell containing the GFF3 header: `ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein DnaA;gene_id=LOOC260_100010;logic_name=ena;version=1`.

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls
hoge1.gtf hoge.bam hoge.gff3
iu@bielinux[mac_share] htseq-count -t gene -i Name -f bam hoge.bam
hoge.gff3 > output_GFF3_gene.txt
Error occurred when processing GFF file (line 18 of file hoge.gff3):
Feature gene:LOOC260_100030 does not contain a 'Name' attribute
[Exception type: ValueError, raised in count.py:53]
iu@bielinux[mac_share] █
```

Contents

■ カウント情報取得の続き

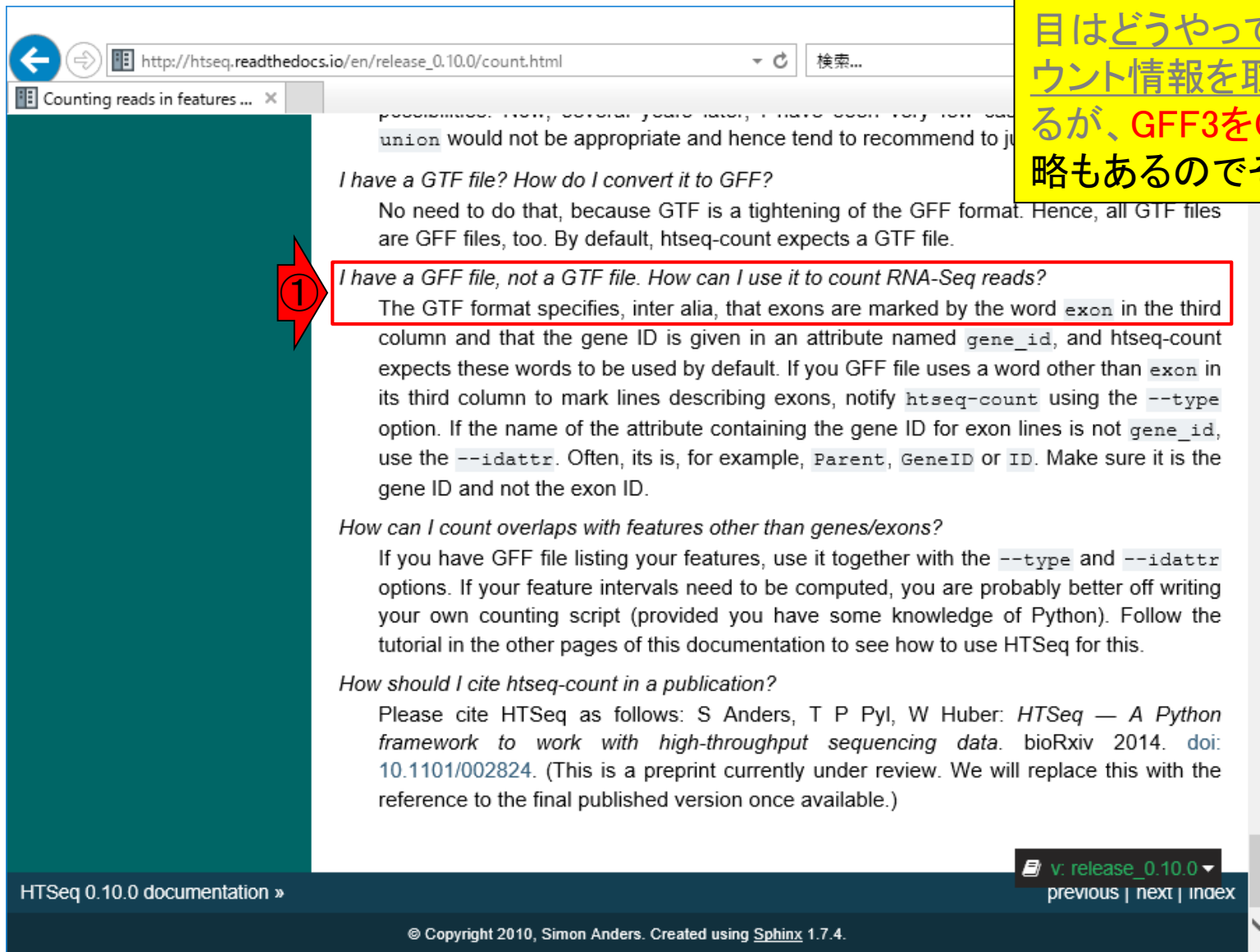
- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

GTFファイルがデフォルト

①あたりの記述内容からも、デフォルトのアノテーションファイルはGFF3ではなくGTFだということがわかる。この項目はどうやってGFF3を読み込ませてカウント情報を取得するかについてであるが、GFF3をGTFに変換するという戦略もあるのでそれを紹介。



http://htseq.readthedocs.io/en/release_0.10.0/count.html

Counting reads in features ... x

problem. Now, several years later, I have seen very few cases where `union` would not be appropriate and hence tend to recommend to just use `union`.

I have a GTF file? How do I convert it to GFF?

No need to do that, because GTF is a tightening of the GFF format. Hence, all GTF files are GFF files, too. By default, `htseq-count` expects a GTF file.

① *I have a GFF file, not a GTF file. How can I use it to count RNA-Seq reads?*

The GTF format specifies, inter alia, that exons are marked by the word `exon` in the third column and that the gene ID is given in an attribute named `gene_id`, and `htseq-count` expects these words to be used by default. If your GFF file uses a word other than `exon` in its third column to mark lines describing exons, notify `htseq-count` using the `--type` option. If the name of the attribute containing the gene ID for exon lines is not `gene_id`, use the `--idattr`. Often, it is, for example, `Parent`, `GeneID` or `ID`. Make sure it is the gene ID and not the exon ID.

How can I count overlaps with features other than genes/exons?

If you have a GFF file listing your features, use it together with the `--type` and `--idattr` options. If your feature intervals need to be computed, you are probably better off writing your own counting script (provided you have some knowledge of Python). Follow the tutorial in the other pages of this documentation to see how to use HTSeq for this.

How should I cite htseq-count in a publication?

Please cite HTSeq as follows: S Anders, T P Pyl, W Huber: *HTSeq — A Python framework to work with high-throughput sequencing data*. bioRxiv 2014. doi: 10.1101/002824. (This is a preprint currently under review. We will replace this with the reference to the final published version once available.)

v. release_0.10.0
previous | next | index

HTSeq 0.10.0 documentation »

© Copyright 2010, Simon Anders. Created using Sphinx 1.7.4.

ファイル形式の変換

②例題1をやってみましょう。③rtracklayerパッケージがなく実行不可能なヒトは、④をダウンロードして眺めましょう。

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページの関連部分には、インストールについての推奨手順 (Windows 2018.0)

なパッケージをインストール版)で自習してください

- イントロ | NGS | 読み込み | [Illuminaの * seq.txt](#) (last modified 2013/06/13)
- イントロ | NGS | 読み込み | [Illuminaの * qseq.txt](#) (last modified 2013/06/17)
- [イントロ | ファイル形式の変換 | について](#) (last modified 2018/05/30) **NEW**
- イントロ | ファイル形式の変換 | [BAM -> BED](#) (last modified 2014/06/21)
- イントロ | ファイル形式の変換 | [FASTQ -> FASTA](#) (last modified 2015/06/15)
- イントロ | ファイル形式の変換 | [Genbank -> FASTA](#) (last modified 2014/03/10)
- イントロ | ファイル形式の変換 | [GFF3 -> GTF](#) (last modified 2018/05/30) **NEW**
- イントロ | ファイル形式の変換 | [qseq -> FASTA](#) (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | [qseq -> FASTA](#) (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | [qseq -> FASTA](#) (last modified 2013/06/17)
- [前処理 | クオリティコントロール |](#)
- [前処理 | クオリティチェック | Quas](#)
- [前処理 | クオリティチェック | grqc](#)

- What's new?
- 「マップ後 | カウン
 - 「イントロ | ファイル
 - 「[H29年度NGSハ](#)
 - [Silhouetteスコアの](#)

イントロ | ファイル形式の変換 | GFF3 -> GTF **NEW**

GFFはGeneral Feature Formatの略で、version 3とversion 2があります。私の認識では、version 3がGFF3で、version 2がGTFです。しかしながら、[GFF version 2を少し改良したものがGTF形式](#)という意見もあるようです。ここでは、[rtracklayer](#)パッケージを用いて、GFF3形式ファイルを読み込んでGTF形式(GFF version 2)で出力するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

② 1. [Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)の場合:

出力ファイルは [hoge1.gtf](#) です

```
in_f <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"
out_f <- "hoge1.gtf" #出力ファイル名を指定してout_fに格納

#必要なパッケージをロード
library(rtracklayer) #パッケージの読み込み

#本番
export(import(in_f), out_f, format="gtf") #GTFに変換
```


①hoge1.gtfをExcelで眺めたところ。②変換前のGFF3ファイルとは確かに形式が異なっていることが分かる。

hoge1.gtf

Excel title bar: hoge1.gtf 保存しました

Excel ribbon: ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示

Formula bar: ID "gene:LOOC260_100010"; Name "dnaA"; biotype "protein_coding"; description "chromosomal replication initiation protein DnaA"; gene_id "LOOC260_100010"; logic_name "ena"; version "1";

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version	2													
2	##source-version	rtracklayer	1.38.3												
3	##date	2018-05-30													
4	Chromosome	ena	gene	360	1676	.	+	.	ID "gene:LOOC260_100010"; Name "dnaA"; biotype "prot						
5	Chromosome	ena	transcript	360	1676	.	+	.	ID "transcript:BAP84581"; Name "dnaA-1"; biotype "prote						
6	Chromosome	ena	exon	360	1676	.	+	.	Name "BAP84581-1"; version "1"; Parent "transcript:BAF						
7	Chromosome	ena	CDS	360	1676	.	+	0	ID "CDS:BAP84581"; Parent "transcript:BAP84581"; prote						
8	Chromosome	ena	gene	1852	2991	.	+	.	ID "gene:LOOC260_100020"; Name "dnaN"; biotype "prot						
9	Chromosome	ena	transcript	1852	2991	.	+	.	ID "transcript:BAP84582"; Name "dnaN-1"; biotype "prote						
10	Chromosome	ena	exon	1852	2991	.	+	.	Name "BAP84582-1"; version "1"; Parent "transcript:BAF						
11	Chromosome	ena	CDS	1852	2991	.	+	0	ID "CDS:BAP84582"; Parent "transcript:BAP84582"; prote						
12	Chromosome	ena	gene	3233	3457	.	+	.	ID "gene:LOOC260_100030"; biotype "protein_coding"; de						
13	Chromosome	ena	transcript	3233	3457	.	+	.	ID "transcript:BAP84583"; biotype "protein_coding"; versi						
14	Chromosome	ena	exon	3233	3457	.	+	.	Name "BAP84583-1"; version "1"; Parent "transcript:BAF						
15	Chromosome	ena	CDS	3233	3457	.	+	0	ID "CDS:BAP84583"; Parent "transcript:BAP84583"; prote						
16	Chromosome	ena	gene	3467	4588	.	+	.	ID "gene:LOOC260_100040"; Name "recF"; biotype "prote						
17	Chromosome	ena	transcript	3467	4588	.	+	.	ID "transcript:BAP84584"; Name "recF-1"; biotype "prote						

Excel status bar: 準備完了 100%

①hoge1.gtfをExcelで眺めたところ。②変換前のGFF3ファイルとは確かに形式が異なっていることが分かる。

hoge.gtf3



hoge.gff3

保存しました

サインイン

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 実行したい作業を入力してください

共有

18 X ✓ fx

ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein DnaA;gene_id=LOOC260_100010;logic_name=ena;version=1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	##gff-version	3													
2	##sequence-region	Chromosome	360	2277853											
3	#!genome-build	European Nucleotide Archive	ASM82939v1												
4	#!genome-version	GCA_000829395.1													
5	#!genome-date	2014-11													
6	#!genome-build-accession	GCA_000829395.1													
7	#!genebuild-last-updated	2014-11													
8	Chromosome	ena	gene	360	1676	.	+	.	ID=gene:LOOC260_100010;Name=dnaA;biotype=protein_coding;description=chromosomal replication initiation protein DnaA;gene_id=LOOC260_100010;logic_name=ena;version=1						
9	Chromosome	ena	transcript	360	1676	.	+	.	ID=transcript:BAP84581;Parent=gene:LOOC260_100010						
10	Chromosome	ena	exon	360	1676	.	+	.	Parent=transcript:BAP84581;Name=BAP84581-1;constitutive						
11	Chromosome	ena	CDS	360	1676	.	+	0	ID=CDS:BAP84581;Parent=transcript:BAP84581;protein_coding						
12	###														
13	Chromosome	ena	gene	1852	2991	.	+	.	ID=gene:LOOC260_100020;Name=dnaN;biotype=protein_coding;description=chromosomal replication initiation protein DnaN;gene_id=LOOC260_100020;logic_name=ena;version=1						
14	Chromosome	ena	transcript	1852	2991	.	+	.	ID=transcript:BAP84582;Parent=gene:LOOC260_100020						
15	Chromosome	ena	exon	1852	2991	.	+	.	Parent=transcript:BAP84582;Name=BAP84582-1;constitutive						
16	Chromosome	ena	CDS	1852	2991	.	+	0	ID=CDS:BAP84582;Parent=transcript:BAP84582;protein_coding						

hoge



準備完了

100%

GTFでカウント情報取得

②の例題5~8が、カウント情報取得時にGTFファイルを指定するやり方です。

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブ
フリーソフト

- [マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/29) **NEW**
- [マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq\(Anders 2015\)](#) (last modified 2018/05/30) **NEW**
- [マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション無 | QuasR\(Gaidatzis 2015\)](#) (last modified 2018/05/26) **NEW**
- [マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション有 | QuasR\(Gaidatzis 2015\)](#) (last modified 2016/02/13)
- [マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション無 | QuasR\(Gaidatzis 2015\)](#) (last modified 2015/07/02)

What's new

- 「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders 2015)」
- 「イントロ | ファイル形式の変換 | GFF3 -> GTF」
- 「H29年 | 正規化 | 基

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) **② W**

[HTSeq](#)というPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample_RNAseq4_3b6c652a602a.bam](#))を利用します。これは、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47](#)のFASTA形式ファイル([sample_RNAseq4_fa](#))です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル([Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。

対応するGFF3形式のアノテーションファイルは[Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge1.gtf](#))は、「イントロ | ファイル形式の変換 | [GFF3 -> GTF](#)」の例題1で作成したものです。また、[sample_RNAseq4_3b6c652a602a.bam](#)も長いので、[hoge.bam](#)として取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature IDを指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

GTFでカウント情報取

例題5~8です。それっぽい結果が得られているので、おそらくこれで大丈夫。無責任な書き方に思われるかもしれませんが、フリーソフト(HTSeq含む)は基本無保証です。ここで利用している乳酸菌のGFF3やGTFでうまく動いても、他の生物種でうまくいくとも限りません。そういうものです。

5. GTFでgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGTF形式であるという前提です。[hoge1.gtf](#)の3列目のgeneで([gene_id](#)の代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なファイルは[output GTF gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge1.gtf > output_GTF_gene.txt
```

6. GTFでtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGTF形式であるという前提です。[hoge1.gtf](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature IDを指定([transcript_id](#)の代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GTF transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge1.gtf > output_GTF_transcript.txt
```

7. GTFでexonレベルのカウントデータを取得する場合:

[hoge1.gtf](#)の3列目のexonでレベル指定、9列目のexon_idでfeature IDを指定([exon_id](#)の代わりにParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GTF exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge1.gtf > output_GTF_exon.txt
```

8. GTFでCDSレベルのカウントデータを取得する場合:

アノテーションファイルがGTF形式であるという前提です。[hoge1.gtf](#)の3列目のCDSでレベル指定、9列目のIDでfeature IDを指定(IDの代わりにprotein_idやParentでもOK)ですが、[protein_id](#)がとちょっと変)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output GTF CDS.txt](#)です。2,194 CDSsですね。

```
htseq-count -t CDS -i ID -f bam hoge.bam hoge1.gtf > output_GTF_CDS.txt
```


Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イントロ、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

おさらい

①例題1の③実行結果ファイル(output_GFF3_gene.txt)の、
④最初の7行と⑤最後の7行。⑥が2,194行目に相当します。

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | HTSeq(Anders_2015) NEW

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015)」の例題10を実行して得られたマッピング結果(sample_RNAseq4_3b6c652a602a.bam)を利用します。これは、Bowtieをデフォルトオプションで実行したものです。マップする側のファイルは、サンプルデータ47のFASTA形式ファイル(sample_RNAseq4.fa)です。マップされる側のファイルは、Ensembl Bacteriaから提供されているLactobacillus casei 12Aのゲノム配列ファイル(Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa)です。対応するGFF3形式のアノテーションファイルはLactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.gff3ですが、ファイル名が長いと見づらいので、hoge.gff3として取り扱います。対応するGTF形式のアノテーションファイル(hoge.gtf)の変換 | GFF3 -> GTFの例題1で作成したものです。また、sample_RNAseq4_3b6c652a602a.bamも長いので取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

① アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のgeneでレベル指定、9列目のgene_idでfeature指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_gene.txtです。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

② アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_transcript.txtです。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

③ アノテーションファイルがGFF3形式であるという前提です。hoge.gff3の3列目のexonでレベル指定、9列目のexon_idでfeature指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル(hoge.bam)なので-f bamとしています(SAMの場合はsam)。出力ファイルはoutput_GFF3_exon.txtです。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0

LOOC260_122680	0
LOOC260_122690	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

カウントデータ

マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション

①赤枠部分が**カウントデータ**と呼ばれるもの。feature(この場合はgene)領域上にリードの一部が重なっていればカウント数を1ずつ増やして作成するデータなので、そのように呼ばれる。

HTSeqというPythonプログラムを用いてカウント情報を得るやり方を示します。ここでは、「マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#)」の例題10を実行して得られたマッピング結果([sample RNAseq4 3b6c652a602a.bam](#))は、[Bowtie](#)をデフォルトオプションで実行したものです。マップする側のファイルは、[サンプルデータ47のFASTA形式ファイル\(sample RNAseq4.fa\)](#)です。マップされる側のファイルは、[Ensembl Bacteria](#)から提供されている [Lactobacillus casei 12A](#)のゲノム配列ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa](#))です。対応するGFF3形式のアノテーションファイルは[Lactobacillus hokkaidonensis jcm 18461.GCA_000829395.1.30.chromosome.gff3](#)ですが、ファイル名が長いと見づらいので、[hoge.gff3](#)として取り扱います。対応するGTF形式のアノテーションファイル([hoge.gtf](#))は、[GTF形式の変換 | GFF3 -> GTF](#)の例題1で作成したものです。また、[sample RNAseq4 3b6c652a602a.bam](#)も長いので取り扱います。

1. GFF3でgeneレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のgeneでレベル指定、9列目のgene_idでfeature指定 (gene_idの代わりにIDでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_gene.txt](#)です。2,194 genesですね。

```
htseq-count -t gene -i gene_id -f bam hoge.bam hoge.gff3 > output_GFF3_gene.txt
```

2. GFF3でtranscriptレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のtranscriptでレベル指定、9列目のtranscript_idでfeature指定 (transcript_idの代わりにIDやParentでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_transcript.txt](#)です。2,250 transcriptsですね。

```
htseq-count -t transcript -i transcript_id -f bam hoge.bam hoge.gff3 > output_GFF3_transcript.txt
```

3. GFF3でexonレベルのカウントデータを取得する場合:

アノテーションファイルがGFF3形式であるという前提です。[hoge.gff3](#)の3列目のexonでレベル指定、9列目のexon_idでfeature指定 (exon_idの代わりにParentやNameでもOK)しています。マッピング結果がBAMファイル([hoge.bam](#))なので-f bamとしています(SAMの場合はsam)。出力ファイルは[output_GFF3_exon.txt](#)です。2,262 exonsですね。

```
htseq-count -t exon -i exon_id -f bam hoge.bam hoge.gff3 > output_GFF3_exon.txt
```

LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

ベクトルではなく行列


通常は、①のような1つのサンプル(1つのRNA-seqリードファイル)のカウントデータのみを取り扱うことはない。①だと、ただの数値ベクトル。



LOOC260_100010	2
LOOC260_100020	5
LOOC260_100030	3
LOOC260_100040	0
LOOC260_100050	0
LOOC260_100060	0
LOOC260_100070	0
LOOC260_122680	0
LOOC260_122690	0
__no_feature	0
__ambiguous	1
__too_low_aQual	0
__not_aligned	0
__alignment_not_unique	0

ベクトルではなく行列

最もシンプルな実験デザインとしては、2つの条件間比較。geneレベルのカウントデータの場合は、①A vs. B間で発現の異なる遺伝子 (Differentially Expressed Genes; DEGs) を調べるのが一般的。



	条件A	条件B
LOOC260_100010	2	
LOOC260_100020	5	
LOOC260_100030	3	
LOOC260_100040	0	
LOOC260_100050	0	
LOOC260_100060	0	
LOOC260_100070	0	
...		
...		
LOOC260_122680	0	
LOOC260_122690	0	

ベクトルではなく行列

最もシンプルな実験デザインとしては、2つの条件間比較。geneレベルのカウントデータの場合は、①A vs. ②B間で発現の異なる遺伝子(Differentially Expressed Genes; DEGs)を調べるのが一般的。通常は、同一グループ(or 同一群 or 同一条件)内のバラツキを評価するため、反復データを取得する。この例は3反復

	①			②		
	A1	A2	A3	B1	B2	B3
LOOC260_100010						
LOOC260_100020						
LOOC260_100030						
LOOC260_100040						
LOOC260_100050						
LOOC260_100060						
LOOC260_100070						
...						
...						
LOOC260_122680						
LOOC260_122690						

実際のカウントデータ

①サンプルデータの、②例題41をコピーで実行した結果が、③20,689 genes × 36 samplesのカウントデータファイル (sample_blekhman_36.txt)。これをコピー実行しても作業ディレクトリ上に④入力ファイルがないのでエラーになります。

- (削除予定)個別パッケージのインストール (last modified 2015/02/20)
- 基本的な利用法 (last modified 2015/04/03)
- サンプルデータ ① (last modified 2015/06/15) **NEW**
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | NGSハンズオン
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 漢語訳
- 書籍
- 書籍
- 書籍

サンプルデータ **NEW**

1. ② 41. Blekhman et al., *Genome Res.*, 2010のリアルカウントデータです。Supplementary Table1で提供されているエクセルファイル (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls>; 約4.3MB)からカウントデータのみ抽出し、きれいに整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesのカウントデータ(sample_blekhman_36.txt)です。実験デザインの詳細はFigure S1中に描かれていますが、ヒト(Homo Sapiens; HS)、チンパンジー(Pan troglodytes; PT)、アカゲザル(Rhesus macaque; RM)の3種類の生物種の肝臓サンプル(liver sample)の比較を行っています。生物種ごとにオス3個体メス3個体の計6個体使われており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を見積もるべく各個体は2つに分割されてデータが取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サンプル、アカゲザル12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトのメス(HSF1, HSF2, HSF3)」, 「ヒトのオス(HSM1, HSM2, HSM3)」, 「チンパンジーのメス(PTF1, PTF2, PTF3)」, 「チンパンジーのオス(PTM1, PTM2, PTM3)」, 「アカゲザルのメス(RMF1, RMF2, RMF3)」, 「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で並び替えたものをファイルに保存しています。もう少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありませんので、ここではアドホック(その場しのぎ、の意味)な手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を把握しているという前提です。尚、「R1L4.HSF1」と「R4L2.HSF1」が「HSF1というヒトのメス一個体のtechnical replicates」であることは列名や文脈から読み解けます。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls" #入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls" #出力ファイル名を指定してout_fに格納
out_f <- "sample_blekhman_36.txt"
```

#入力ファイルの読み込み

```
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind( #必要な列名を取得したい列の順番で結合した結果をdataに格納
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6, hoge$R6L4.PTM3,
  hoge$R1L7, hoge$R4L7.RMF3,
  hoge$R1L3, hoge$R4L3.RMM3)
```

Blekhman et al., *Genome Res.*, 20: 180-189, 2010

実際のカウントデータ

①から20,689 genes × 36 samplesのカウントデータファイル(sample_blekhman_36.txt)。をダウンロードしてもよいが、②からもダウンロードできます。

- (削除予定)個別パッケージのインストール (last modified 2015/02/20)
- 基本的な利用法 (last modified 2015/04/03)
- サンプルデータ (last modified 2015/06/15) **NEW**
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | [NGSハンズオン](#)
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | [NGSハンズオン](#)

サンプルデータ **NEW**

- 書籍
- 書籍
- 書籍

41. Blekhman et al., *Genome Res.*, 2010のリアルカウントデータです。Supplementary Table1で提供されているエクセルファイル (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls>; 約4.3MB)からカウントデータのみ抽出し、きれいに整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesのカウントデータ(sample_blekhman_36.txt)①。実験デザインの詳細はFigure S1中に描かれていますが、ヒト(Homo Sapiens; HS)、チンパンジー(Pan troglodytes; PT)、アカゲザル(Macaca mulatta; RM)の3種類の生物種の肝臓サンプル(liver sample)の比較を行っています。生物種ごとにオス3個体メス3個体の計6個体使用されており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を見積もるべく各個体は2つに分割されてデータが取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サンプル、アカゲザル12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトのメス(HSF1, HSF2, HSF3)」, 「ヒトのオス(HSM1, HSM2, HSM3)」, 「チンパンジーのメス(PTF1, PTF2, PTF3)」, 「チンパンジーのオス(PTM1, PTM2, PTM3)」, 「アカゲザルのメス(RMF1, RMF2, RMF3)」, 「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありませんので手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を照らし合わせ、"R1L4.HSF1"と"R4L2.HSF1"が「HSF1というヒトのメス一個体のtechnical replicates

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls"
in_f <- "suppTable1.xls" #入力ファイル名を指定してin_f
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_f
```

```
#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="\"", as.is=TRUE)
dim(hoge) #行数と列数を表示
```

```
#サブセットの取得
data <- cbind( #必要な列名の情報を取得した
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R4L4.PTM1, hoge$R2L7.PTM2, hoge$R3L7.PTM3, hoge$R6L4.PTM3,
  hoge$R1L7.RMF1, hoge$R4L7.RMF1, hoge$R2L8.RMF2, hoge$R4L7.RMF3,
  hoge$R1L3.RMM1, hoge$R4L3.RMM1, hoge$R2L9.RMM2, hoge$R4L3.RMM3)
```

講義日程 (平成30年度)

1. 平成30年06月12日 (PC使用)
講義資料PDF
(Rで)塩基配列解析
QuasR : Gaidatzis et al., *Bioinformatics*, 2015
HTSeq : Anders et al., *Bioinformatics*, 2015
hoge10.txt
htseq-countのページ
hoge1.gtf
sample_blekhman_36.txt ②
2. 平成30年06月19日 (PC使用)
3. 平成30年06月26日 (PC使用)
4. 平成30年07月03日 (PC使用)

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イン트로、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

EXCELで概観

sample_blekhman_36.txtをExcelで眺めるとこんな感じ。①のサンプルで考えると、②はENSG00000000971という遺伝子領域上に2,262リードマップされたことを表す。③はENSG00000001460の遺伝子領域上に3リードマップされたことを表す。もしこの2つの配列長が同じなら、マップされたリード数が多い前者②の発現量が高いという理解でよい。

	A	C	D	E	F	G	H	I	J
1	R1 L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3	R1 L1.HSM1	R5L2.HSM1	R2L3
2	ENSG00000000003	172	157	147	153	78	90	60	61
3	ENSG00000000005	0	0	0	0	0	0	0	0
4	ENSG00000000419	36	45	26	35	16	40	17	22
5	ENSG00000000457	41	50	28	34	34	42	50	64
6	ENSG00000000460	3	3	8	9	7	5	9	6
7	ENSG00000000938	23	21	30	35	112	98	32	41
8	ENSG00000000971	2262	2503	3473	3752	1665	1740	1726	1874
9	ENSG00000001036	155	142	118	133	79	110	99	101
10	ENSG00000001084	323	307	377	360	151	155	155	181
11	ENSG00000001167	19	17	15	15	16	20	13	16
12	ENSG00000001460	3	0	0	1	1	4	0	1
13	ENSG00000001461	25	24	22	15	14	20	13	15
14	ENSG00000001497	59	58	46	47	46	43	39	41
15	ENSG00000001561	22	26	23	27	28	25	29	33
16	ENSG00000001617	30	34	24	27	77	73	40	30
17	ENSG00000001626	9	3	12	32	37	33	24	19

データの正規化

①のサンプル内で、②は③より $2262/3 = 754$ 倍高発現と評価してはいけない。発現量の大小関係を比較したい場合は、長さで補正する必要がある。このあたりは④参考書のp132-137で述べている。

sample_blekhman

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 アドイン 門田幸二

A1 : fx

	A	B	C	D	E	F	G	H	I	J
1		R1 L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3	R1 L1.HSM1	R5L2.HSM1	R2L3
2	ENSG000000000003	172	157	147	153	78	90	60	61	2
3	ENSG000000000005	0	0	0	0	0	0	0	0	
4	ENSG000000000419	36	45	26	35	16	40	17	22	
5	ENSG000000000457	41	50	28	34	34	42	50	64	
6	ENSG000000000460	3	3	8	9	7	5	9	6	
7	ENSG000000000938	23	21	30	35	112	98	32	41	
8	ENSG000000000971	2262	2503	3473	3752	1665	1740	1726	1874	32
9	ENSG000000001036	155	142	119	122	79	110	99	101	
10	ENSG000000001084	323	307	3						
11	ENSG000000001167	19	17							
12	ENSG000000001460	3	0							
13	ENSG000000001461	25	24							
14	ENSG000000001497	59	58							
15	ENSG000000001561	22	26							
16	ENSG000000001617	30	34							
17	ENSG000000001626	9	3							

sample_blekhman_36 (+)

準備完了

- 書籍 [トランスクリプトーム解析](#) について (last modified 2014/05/12)
- 書籍 [トランスクリプトーム解析](#) [2.3.1 RNA-seqデータ\(FASTQファイル\)](#) (last modified 2016/02/09)
- 書籍 [トランスクリプトーム解析](#) [2.3.2 リファレンス配列](#) (last modified 2014/04/16)
- 書籍 [トランスクリプトーム解析](#) [2.3.3 アンテーション情報](#) (last modified 2014/04/17)
- 書籍 [トランスクリプトーム解析](#) [2.3.4 マッピング\(準備\)](#) (last modified 2014/06/20)
- 書籍 [トランスクリプトーム解析](#) [2.3.5 マッピング\(本番\)](#) (last modified 2014/06/21)
- 書籍 [トランスクリプトーム解析](#) [2.3.6 カウントデータ取得](#) (last modified 2016/02/09)
- 書籍 [トランスクリプトーム解析](#) [3.3.1 解析目的別留意点](#) (last modified 2014/04/20)
- 書籍 [トランスクリプトーム解析](#) [3.3.2 データの正規化\(基礎編\)](#) (last modified 2014/06/23)
- 書籍 [トランスクリプトーム解析](#) [3.3.3 クラスターリング](#) (last modified 2014/04/20)
- 書籍 [トランスクリプトーム解析](#) [3.3.4 各種プロット](#) (last modified 2014/04/27)
- 書籍 [トランスクリプトーム解析](#) [4.3.1 シミュレーションデータ\(負の二項分布\)](#) (last modified 2014/04/27)
- 書籍 [トランスクリプトーム解析](#) [4.3.2 データの正規化\(応用編\)](#) (last modified 2014/04/27)
- 書籍 [トランスクリプトーム解析](#) [4.3.3 2群間比較](#) (last modified 2014/04/28)
- 書籍 [トランスクリプトーム解析](#) [4.3.4 他の実験デザイン\(3群間\)](#) (last modified 2014/04/28)

データの正規化

例えば、②と③の配列長がそれぞれ④3000塩基、⑤500塩基だったと仮定すると、②は③に対して $3,000/500 = 6$ 倍長いので、その分を補正してやる必要がある。⑥様々な表現方法があるが、発現量の比率(②/③)で考えると125.6667倍というのは不変

The screenshot shows an RStudio spreadsheet with columns A through J. Red arrows and circles with numbers 1-6 highlight specific data points and calculations. An R Console window is open in the bottom right, showing the following commands and results:

```

> 2262/3
[1] 754
> (2262/6)/3
[1] 125.6667
> (2262/3000)/(3/500)
[1] 125.6667

```

	A	B	C	D	E	F	G	H	I	J
1		R1 L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3	R1 L1.HSM1	R5L2.HSM1	R2L3
2	ENSG000000000003	172	157	147	153	78	90	60	61	2
3	ENSG000000000005	0	0	0	0	0	0	0	0	
4	ENSG000000000419	36	45	26	35	16	40	17	22	
5	ENSG000000000457	41	50	28	34	34	42	50	64	
6	ENSG000000000460	3	3	8	9	7	5	9	6	
7	ENSG000000000938	23	21	30	35	112	98	32	41	
8	ENSG000000000971	2262	2503	3473	3752	1665	1740	1726	1874	32
9	ENSG00000001036	155	142	1	133	79	110	99	101	
10	ENSG00000001084	323	307	377	360	151	155	155	181	4
11	ENSG00000001167	19	17	15	15	16	20	13	16	
12	ENSG00000001460	3	0	0	1	1				
13	ENSG00000001461		24	22	15	14				
14	ENSG00000001497	59	58	46	47	46				
15	ENSG00000001561	22	26	23	27	28				
16	ENSG00000001617	30	34	24	27	77				
17	ENSG00000001626	9	3	12	32	37				

RPK補正のイントロ

④は「マップされたリード数(生のカウント数) × 1 / 配列長」に相当する。得られる数値は、塩基あたりのリード数(Reads per one base)ともいえる。これが長さ補正の基本形であるが、得られる数値(0.754や0.006)が小さすぎるのが難点

sample_blekhman_36

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 アド

A1 :

	A	C	D	E	F	G	H	I	J	
1		R1 L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3	R1 L1.HSM1	R5L2.HSM1	R2L3
2	ENSG000000000003	172	157	147	153	78	90	60	61	2
3	ENSG000000000005	0	0	0	0	0	0	0	0	
4	ENSG000000000419	36	45	26	35	16	40	17	22	
5	ENSG000000000457	41	50	28	34	34	42	50	64	
6	ENSG000000000460	3	3	8	9	7	5	9	6	
7	ENSG000000000938	23	21	30	35	112				
8	ENSG000000000971	2262	2503	3473	3752	1665				
9	ENSG00000001036	155	142	118	133	79				
10	ENSG00000001084	323	307	377	360	151				
11	ENSG00000001167	19	17	15	15	16				
12	ENSG00000001460	3	0	0	1	1				
13	ENSG00000001461	25	24	22	15	14				
14	ENSG00000001497	59	58	46	47	46				
15	ENSG00000001561	22	26	23	27	28				
16	ENSG00000001617	30	34	24	27	77				
17	ENSG00000001626	9	3	12	32	37				

sample_blekhman_36

準備完了

R Console

```

> (2262/6)/3
[1] 125.6667
> (2262/3000)/(3/500)
[1] 125.6667
> 2262/3000
[1] 0.754
> 3/500
[1] 0.006
> 2262*(1000/3000)
[1] 754
> 3*(1000/500)
[1] 6
> |

```

RPK補正

④は「マップされたリード数(生のカウント数) × 1000 / 配列長」に相当する。得られる数値は、1000塩基あたりのリード数(Reads per one kilobase; RPK)ともいえる。配列長の異なる遺伝子間の発現レベルの大小関係を平等に比較すべく、「遺伝子が1000 bpだったときのリード数」とするのがRPKの考え方。RPK補正後の値は②が754、③が6となる

sample_blekhman_36

	A	C	D
1	R1 L4.HSF1	R4L2.HSF1	R2L7.HSF2
2	ENSG000000000003	172	157
3	ENSG000000000005	0	0
4	ENSG000000000419	36	45
5	ENSG000000000457	41	50
6	ENSG000000000460	3	3
7	ENSG000000000938	23	21
8	ENSG000000000971	2262	2503
9	ENSG000000001036	155	142
10	ENSG000000001084	323	307
11	ENSG000000001167	19	17
12	ENSG000000001460	3	0
13	ENSG000000001461	25	24
14	ENSG000000001497	59	58
15	ENSG000000001561	22	26
16	ENSG000000001617	30	34
17	ENSG000000001626	9	3

sample_blekhman_36

準備完了

R Console

```

> (2262/6) / 3
[1] 125.6667
> (2262/3000) / (3/500)
[1] 125.6667
> 2262/3000
[1] 0.754
> 3/500
[1] 0.006
> 2262 * (1000/3000)
[1] 754
> 3 * (1000/500)
[1] 6

```

RPK補正

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブページのR関連部分は、[インストール](#)についての推奨手順 ([Windows2018.0](#)なパッケージをインストール済みであるという前提で記述しています。初心者の方は[基本版](#))で自習してください。本ウェブページを体系的にまとめた[書籍](#)もあります。(2015/04/03)

①RPKの例題はこちら。当然配列長の情報が必要です。配列長補正が必要な局面は、同一サンプル内で異なる遺伝子間の発現レベルの大小関係を知りたい場合、です。

What's new?

- 「マップ後 | カウント」
- 「イントロ | ファイル」
- 「[H29年度NGSハン](#)」
- Silhouetteスコアの解

- マップ後 | カウント 情報取得 | トランスクリプトーム | [BEDファイルから](#) (last modified 2014/06/21)
- マップ後 | [配列長とカウント 数の関係](#) (last modified 2015/07/03)
- [正規化](#) | [について](#) (last modified 2014/06/22)
- 正規化 | 基礎 | [RPK or CPK \(配列長補正\)](#) ① (last modified 2015/07/04)
- 正規化 | 基礎 | [RPM or CPM \(総リード 数補正\)](#) (last modified 2016/05/12)
- 正規化 | 基礎 | [RPKM](#) (last modified 2015/07/04)
- 正規化 | サンプル内 | [EDASeq\(Risso 2011\)](#) (last modified 2013/06/24)
- 正規化 | サンプル内 | [RNASeqBias\(Zheng 2011\)](#) (last modified 2013/06/24)
- [正規化](#) | [サンプル間](#) | [について](#) (last modified 2015/11/10)

正規化 | 基礎 | RPK or CPK (配列長補正)

ここでは、遺伝子(転写物)ごとのリード数を「配列長が1000 bp (one kilobase)だったときのリード数: Reads per kilobase (RPK)」に変換するやり方を示します。「リード数 = カウント 数」なのでReadsのところをCountsに置き換えた表現(Counts per kilobase; CPK)もときどき見受けられます。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. 配列長とカウント 情報を含むファイル([sample length count.txt](#))の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント 数からなるファイルです。基本形です。

```

in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param <- 1000 # 「Reads per X」のXの値を指定(デフォルトはRPKなので1000)

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイル
head(data) #確認してるだけです

```

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イントロ、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

RPM補正

①20,689 genes × 36 samplesのカウントデータファイル(sample_blekhman_36.txt)に対してRPM補正を実行するのは、②例題10

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブ
なバック
版)で自

- マップ後 | カウント情報取得 | トランスクリプトーム | [BEDファイルから](#) (last modified 2014/06/21)
- マップ後 | [配列長とカウント数の関係](#) (last modified 2015/07/03)
- [正規化](#) | [正規化について](#) (last modified 2014/06/22)
- 正規化 | 基礎 | [RPK or CPK \(配列長補正\)](#) (last modified 2015/07/04)
- 正規化 | 基礎 | [RPM or CPM \(総リード数補正\)](#) (last modified 2016/05/12)
- 正規化 | 基礎 | [RPKM](#) (last modified 2015/07/04)

What's

- 「マッ
- 「イン
- 「H29
- Silho

- 正規化 | サンプル内
- 正規化 | サンプル内
- 正規化 | サンプル間
- 正規化 | サンプル間
- 正規化 | サンプル間
- 正規化 | サンプル間
- 正規化 | サンプル間

正規化 | 基礎 | RPM or CPM (総リード数補正) NEW

カウントデータファイルを読み込んで、転写物ごとのリード数を「総リード数が100万 (million)だったときのリード数: Reads per million (RPM)」に変換するやり方を示します。「リード数 = カウント数」なのでReadsのところをCountsに置き換えて表現(Counts per million: CPM)もときどき見受けられます。

② 10. サンプルデータ①の20,689 genes × 36 samplesのカウントデータ(sample_blekhman_36.txt)の場合:

1. 配列長とカウント情報を

1-3列目がそれぞれ、gen

```
in_f <- "sample_le  
out_f <- "hoge1.tx  
param1 <- 1000000
```

```
#入力ファイルの読み込み  
data <- read.table
```

```
in_f <- "sample_blekhman_36.txt" #入力ファイル名を指定してin_fに格納  
out_f <- "hoge10.txt" #出力ファイル名を指定してout_fに格納  
param1 <- 1000000 #補正後の総リード数を指定(RPMにしたい場合はこ  
  
#入力ファイルの読み込み  
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定  
colSums(data) #総リード数を表示  
  
#本番(正規化)  
nf <- param1/colSums(data) #正規化係数を計算した結果をnfiに格納  
data <- sweep(data, 2, nf, "*") #正規化係数を各列に掛けた結果をdataに格納  
colSums(data) #総リード数を表示  
  
#ファイルに保存  
tmp <- cbind(rownames(data), data) #保存したい情報をtmpに格納  
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定
```


RPM補正のイントロ

スライドを見るだけ。サンプル(列)ごとにマップされた総リード数を計算した結果。サンプル間比較の場合には、この総リード数を揃えるのが基本戦略。総リード数を100万(one million)に揃えるのが、RPM (Reads per million)補正

sample_blekhman_36.

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 アドイン

B20692 : *fx* =SUM(B2:B20690)

	A	B	C	D	E	F	G	H	I
20677	ENSG00000221765	0	0	0	0	0	0	0	0
20678	ENSG00000221766	0	0	0	0	0	0	0	0
20679	ENSG00000221767	0	0	0	0	0	0	0	0
20680	ENSG00000221768	0	0	0	0	0	0	0	0
20681	ENSG00000221770	4	2	4	0	2	2	0	0
20682	ENSG00000221771	0	0	0	0	0	0	0	0
20683	ENSG00000221775	0	0	0	0	0	0	0	0
20684	ENSG00000221778	0	0	0	0	0	0	0	0
20685	ENSG00000221781	0	0	0	0	0	0	0	0
20686	ENSG00000221782	0	0	0	0	0	0	0	0
20687	ENSG00000221783	0	0	0	0	0	1	0	0
20688	ENSG00000221784	0	0	0	0	0	0	0	0
20689	ENSG00000221786	0	0	0	0	0	0	0	0
20690	ENSG00000221788	0	0	0	0	0	0	0	0
20691									
20692		1665987	1719125	1620189	1801009	1393867	1450604	1346515	1497738
20693									

sample_blekhman_36

準備完了

100%

colSums関数で、列ごとの総リード数を一気に表示。ExcelとR間で同じ値が得られていることがわかる(①と②)。③RPM補正後のデータで同じ操作を実行すると、全部100万になる(ここはまだ補正前の状態)

RPM補正のイントロ

The screenshot shows an Excel spreadsheet with an R Console window overlaid. The spreadsheet has columns A and B. Column A contains gene IDs (e.g., ENSG00000221765) and column B contains counts (e.g., 0). The R Console window shows the output of the `colSums(data)` function, displaying a matrix of values. Red arrows labeled ①, ②, and ③ point to specific values in both the console and the spreadsheet to illustrate the comparison.

Gene ID (A)	Count (B)	colSums (R Console)
20677	0	R1L4.HSF1
20678	0	1665987
20679	0	R1L1.HSM1
20680	0	1346515
20681	4	R1L2.PTF1
20682	0	2667264
20683	0	R1L6.PTM1
20684	0	1481536
20685	0	R1L7.RMF1
20686	0	2400660
20687	0	R1L3.RMM1
20688	0	2657274
20689	0	
20690	0	
20691		
20692	1665987	1665987
20693		

RPM補正

①入力は、20,689 genes × 36 samplesのカウントデータ。サンプル(列)ごとに総リード数は異なるので、②正規化係数nfは列ごとに異なる。③nfの中身。数値ベクトルnfの要素数は、列数と同じく36。②のnfオブジェクトの中身を見るために、必要な部分までなど自由にコピー実行してよい

10. サンプルデータ41の20,689 genes×36 samplesのカウントデータ

```
in_f <- "sample_blekhman_36.txt"
out_f <- "hoge10.txt"
param1 <- 1000000

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")
colSums(data)

#正規化
nf <- param1/colSums(data)
data <- sweep(data, 2, nf, "*")
colSums(data)

#ファイルに保存
tmp <- cbind(rownames(data), data)
write.table(tmp, out_f, sep="\t")
```

① #入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_fに格納
#補正後の総リード数を指定(RPMにしたい場合はこ)

② #正規化係数を計算した結果をnfに格納
#正規化係数を各列に掛けた結果をdataに格納
#総リード数を表示

③

```
R Console
> nf
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8L1.HSF3 R8L2.HSF3
0.6002448 0.5816913 0.6172119 0.5552443 0.7174286 0.6893680
R1L1.HSM1 R5L2.HSM1 R2L3.HSM2 R4L8.HSM2 R3L6.HSM3 R4L1.HSM3
0.7426579 0.6676735 0.4510122 0.4612559 0.5065271 0.5478332
R1L2.PTF1 R4L4.PTF1 R2L4.PTF2 R6L6.PTF2 R3L7.PTF3 R5L3.PTF3
0.3749160 0.3734449 0.5234500 0.5315103 0.5439882 0.5512928
R1L6.PTM1 R3L3.PTM1 R2L8.PTM2 R4L6.PTM2 R6L2.PTM3 R6L4.PTM3
0.6749752 0.5900791 0.6218372 0.5137394 0.5730042 0.5544605
R1L7.RMF1 R5L1.RMF1 R2L2.RMF2 R5L8.RMF2 R3L4.RMF3 R4L7.RMF3
0.4165521 0.4737527 0.4276368 0.6519304 0.3723486 0.3945404
R1L3.RMM1 R3L8.RMM1 R2L6.RMM2 R5L4.RMM2 R3L1.RMM3 R4L3.RMM3
0.3763255 0.3990517 0.5148546 0.5064568 0.4718103 0.4146441
> |
```

RPM補正

①nfベクトルの1番目の要素である、R1L4.HSF1サンプルの正規化係数(0.6002448)は、②1,000,000 / 1,665,987 = 0.6002448として計算している。ここで、③1,665,987はR1L4.HSF1サンプルの総リード数

10. サンプルデータ41の20,689 genes×36 samplesのカウントデータ(sample

```
in_f <- "sample_blekhman_36.txt"
out_f <- "hoge10.txt"
param1 <- 1000000
```

#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_fに格納
#補正後の総リード数を指定(RPMにしたい場合はこ

#入力ファイルの読み込み

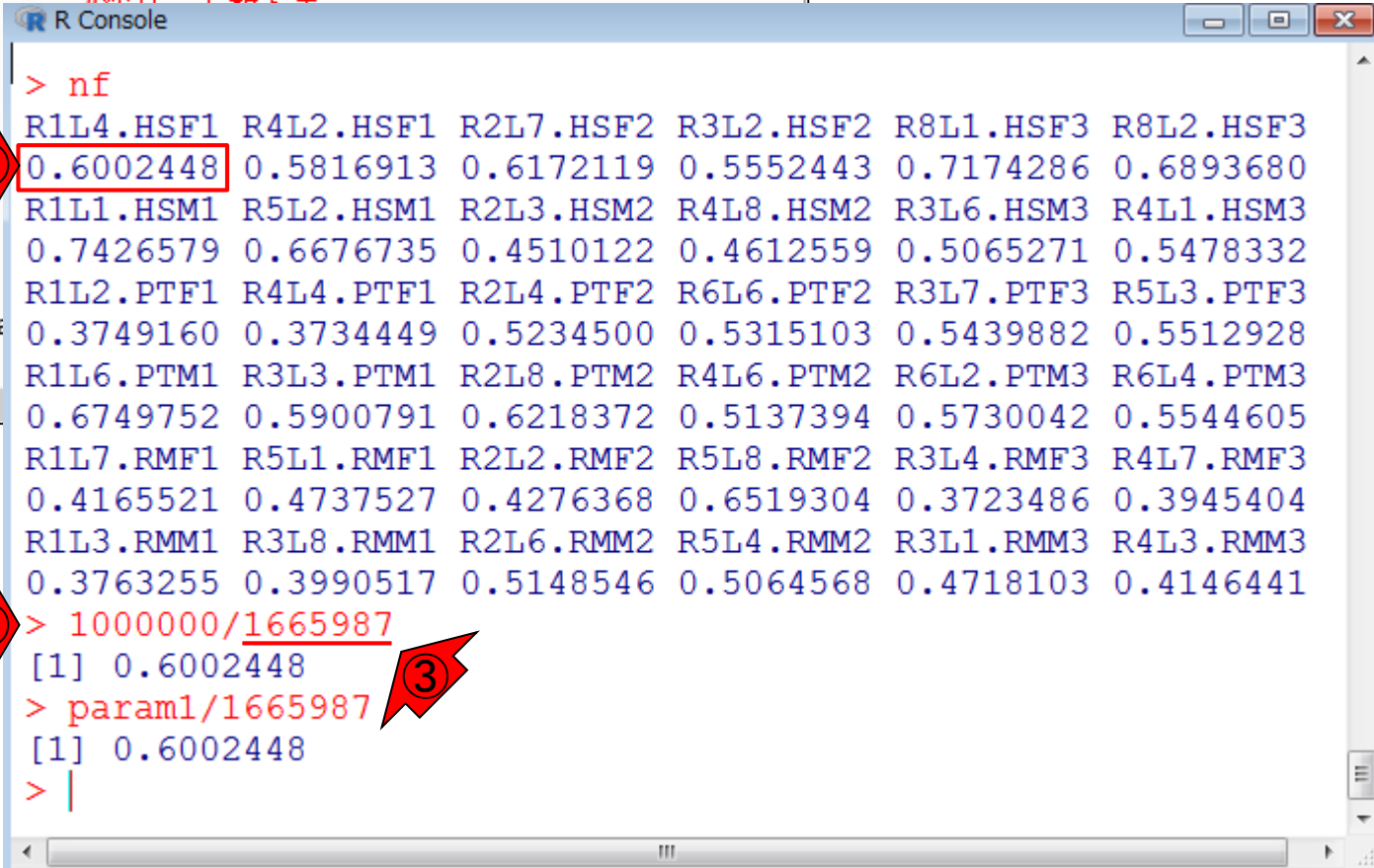
```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定
colSums(data)
```

#本番(正規化)

```
nf <- param1/colSums(data)
data <- sweep(data, 2, nf, "*")
colSums(data)
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data)
write.table(tmp, out_f, sep="\t", e
```



```
R Console
> nf
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8L1.HSF3 R8L2.HSF3
0.6002448 0.5816913 0.6172119 0.5552443 0.7174286 0.6893680
R1L1.HSM1 R5L2.HSM1 R2L3.HSM2 R4L8.HSM2 R3L6.HSM3 R4L1.HSM3
0.7426579 0.6676735 0.4510122 0.4612559 0.5065271 0.5478332
R1L2.PTF1 R4L4.PTF1 R2L4.PTF2 R6L6.PTF2 R3L7.PTF3 R5L3.PTF3
0.3749160 0.3734449 0.5234500 0.5315103 0.5439882 0.5512928
R1L6.PTM1 R3L3.PTM1 R2L8.PTM2 R4L6.PTM2 R6L2.PTM3 R6L4.PTM3
0.6749752 0.5900791 0.6218372 0.5137394 0.5730042 0.5544605
R1L7.RMF1 R5L1.RMF1 R2L2.RMF2 R5L8.RMF2 R3L4.RMF3 R4L7.RMF3
0.4165521 0.4737527 0.4276368 0.6519304 0.3723486 0.3945404
R1L3.RMM1 R3L8.RMM1 R2L6.RMM2 R5L4.RMM2 R3L1.RMM3 R4L3.RMM3
0.3763255 0.3990517 0.5148546 0.5064568 0.4718103 0.4146441
> 1000000/1665987
[1] 0.6002448
> param1/1665987
[1] 0.6002448
> |
```


RPM補正は、①入力ファイル情報に相当するdataの、②各列に対して、③正規化係数nfを、④掛けた結果を、再びdataオブジェクトに格納することで達成

RPM補正

10. サンプルデータ41の20,689 genes×36 samplesのカウントデータ(sample blekhman 36.txt)の場合:

```

in_f <- "sample_blekhman_36.txt"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge10.txt"                 #出力ファイル名を指定してout_fに格納
param1 <- 1000000                     #補正後の総リード数を指定(RPMにしたい場合はこ

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
colSums(data)                         #総リード数を表示

#本番(正規化)
nf <- param1/colSums(data)             #正規化係数を計算した結果をnfに格納
data <- sweep(data, 2, nf, "*")        #正規化係数を各列に掛けた結果をdataに格納
colSums(data)                         #総リード数を表示

#ファイルに保存
tmp <- cbind(row.names(data), data)    #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定

```

RPM補正

RPM補正後のdataオブジェクトに対して、colSums関数で各列の総リード数を表示。全部同じ100万(1e+06)になっていることがわかる。総リード数が揃っているので、サンプル間で大幅に数値が異なるという事態は回避できる。

10. サンプルデータ41の20,689 genes×36 samplesのカウントデータ(sample blek

```
in_f <- "sample_blekhman_36.txt" #入力ファイル名を指定し
out_f <- "hoge10.txt" #出力ファイル名を指定してout_fに格納
param1 <- 1000000 #補正後の総リード数を指定(RPMにしたい場合はこ
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, as.is=TRUE)
colSums(data)
```

#本番(正規化)

```
nf <- param1/colSums(data)
data <- sweep(data, 2, nf, "*")
colSums(data)
```

#ファイルに保

```
tmp <- cbind(rownames(data), data)
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.$
```

R Console

```
> data <- sweep(data, 2, nf, "*") #正規化係数を各列$
> colSums(data) #総リード数を表示
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8L1.HSF3 R8L2.HSF3
1e+06 1e+06 1e+06 1e+06 1e+06 1e+06
R1L1.HSM1 R5L2.HSM1 R2L3.HSM2 R4L8.HSM2 R3L6.HSM3 R4L1.HSM3
1e+06 1e+06 1e+06 1e+06 1e+06 1e+06
R1L2.PTF1 R4L4.PTF1 R2L4.PTF2 R6L6.PTF2 R3L7.PTF3 R5L3.PTF3
1e+06 1e+06 1e+06 1e+06 1e+06 1e+06
R1L6.PTM1 R3L3.PTM1 R2L8.PTM2 R4L6.PTM2 R6L2.PTM3 R6L4.PTM3
1e+06 1e+06 1e+06 1e+06 1e+06 1e+06
R1L7.RMF1 R5L1.RMF1 R2L2.RMF2 R5L8.RMF2 R3L4.RMF3 R4L7.RMF3
1e+06 1e+06 1e+06 1e+06 1e+06 1e+06
R1L3.RMM1 R3L8.RMM1 R2L6.RMM2 R5L4.RMM2 R3L1.RMM3 R4L3.RMM3
1e+06 1e+06 1e+06 1e+06 1e+06 1e+06
>
> #ファイルに保存
> tmp <- cbind(rownames(data), data) #保存したい情報をt$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.$
> |
```

Contents

■ カウント情報取得の続き

- フォローアップ(なぜ365 genesとなったのか?)
- HTSeqでカウント情報取得
 - htseq-countとカウントモード
 - Usage(利用法)の読み解き方、実行(geneレベルカウントデータの取得)
 - 結果の解釈、応用スキルの習得
 - 課題1~3
 - 課題4(-t gene -i Nameとして、gene symbolをfeatureとして使うには)
 - ファイル形式の変換(GFF3 → GTF)

■ データの正規化(RPK, RPM, RPKM/FPKM)

- イントロ、RPK(長さの違いを補正)
- RPM(総リード数の違いを補正)
- RPKM/FPKM(長さ²と総リード数の両方を補正)

RPKM補正

① RPKMは、配列長補正 (RPK) と総リード数補正 (RPM) を組み合わせただけです。これがよく expression level として取り扱われます。今では RPKM ではなく FPKM がよく使われます。

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブ
なバック
版)で自

What's

- 「マッ
- 「イン
- 「H29
- Silho

- マッピング後 | カウント情報取得 | トランスクリプトーム | [BEDファイルから](#) (last modified 2014/06/21)
- マッピング後 | [配列長とカウント数の関係](#) (last modified 2015/07/03)
- [正規化](#) | [正規化について](#) (last modified 2014/06/22)
- 正規化 | 基礎 | [RPK or CPK \(配列長補正\)](#) (last modified 2015/07/04)
- 正規化 | 基礎 | [RPM or FPKM \(総リード数補正\)](#) (last modified 2015/07/04)
- 正規化 | 基礎 | [RPKM](#) (last modified 2015/07/04)
- 正規化 | サンプル内 | [RNASeq\(Risso 2011\)](#) (last modified 2015/07/04)
- 正規化 | サンプル内 | [RNASeqBias\(Zheng 2011\)](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | [正規化 サンプル間について](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | [Upper-quartile\(Bullard 2010\)](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | [Quantile\(Bullard 2010\)](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | 2群間 | 複製あり | [iDEGES](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | 2群間 | 複製あり | [DEGES](#) (last modified 2015/07/04)

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示

#本番(正規化)
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正用)を計算した結果をnf_RPM
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正用)を計算した結果をnf_RPK
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に掛けた結果を元の位置に代入
head(data) #確認してるだけです

#ファイルに保存
tmp <- cbind(rownames(data), data) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定
```

RPKM補正

(Rで)塩基配列解析

(last modified 2018/05/30, since 2010)

このウェブ
なバック
版)で自

What's

- 「マッ
- 「イン
- 「H29
- Silho

- マッピング | カウント情報取得 | トランスクリプトーム | [BEDTools](#)
- マッピング | [配列長とカウント数の関係](#) (last modified 2015/07/04)
- [正規化](#) | [正規化について](#) (last modified 2014/06/22)
- 正規化 | 基礎 | [RPK or CPK \(配列長補正\)](#) (last modified 2015/07/04)
- 正規化 | 基礎 | [RPM or FPKM \(総リード数補正\)](#) (last modified 2015/07/04)
- 正規化 | 基礎 | [RPKM](#) (last modified 2015/07/04)
- 正規化 | サンプル内 | [EdgeSeq\(Risso 2011\)](#) (last modified 2015/07/04)
- 正規化 | サンプル内 | [RNASeqBias\(Zheng 2011\)](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | [正規化 サンプル間について](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | [Upper-quartile\(Bullard 2010\)](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | [Quantile\(Bullard 2010\)](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | 2群間 | 複製あり | [iDEGES](#) (last modified 2015/07/04)
- 正規化 | サンプル間 | 2群間 | 複製あり | [DEGES](#) (last modified 2015/07/04)

これは、昔の超short read時代は「マップされたread数」をベースとしていたが、今はpaired-endデータでリード長も長くなってきたので、paired-endで読む前の断片配列(fragment)がいくつマップされたのかを考えるようになったからです。だからRPKMではなくFPKMです。コンセプトは全く同じですが、呼び方が違うだけという理解で差し支えありません。

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示

#本番(正規化)
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正用)を計算した結果をnf_RPM
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正用)を計算した結果をnf_RPK
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に掛けた結果を元の位置に代入
head(data) #確認してるだけです

#ファイルに保存
tmp <- cbind(rownames(data), data) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定
```


一部をコピペ

①RPKM補正の実例を実データで簡単に示すため、これまでとはフォーマットが異なる。
②赤枠内を、③コピペ実行した結果画面

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけ
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)
```

```
R Console
> #入力ファイルの読み込み
> data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
> head(data) #確認$
      Length Count
NM_203348.1   3543     3
NM_001008737.1 1897    19
NM_001037228.1  537     7
NM_033183.2    886     0
NM_138368.3   4443    56
NM_152833.2   2844    85
> sum(data[,2]) #総リード$
[1] 2385273
>
> |
```

dataオブジェクト

①dataオブジェクトの、②1列目が配列長情報、③2列目がカウント情報。

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmp
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=FALSE)
```

```
R Console
> data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #確認$
> head(data)
      Length Count
NM_203348.1  3543    3
NM_001008737.1 1897   19
NM_001037228.1  537    7
NM_033183.2    886    0
NM_138368.3   4443   56
NM_152833.2   2844   85
> sum(data[,2]) #総リード$
[1] 2385273
>
> |
```

総リード数は約240万

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmp
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)
```

①総リード数は2,385,273。100万に揃える総リード数(RPM)補正の段階で、②のカウント数が半分以下(正確には1/2.385273)になるのだろうと予想する。

```
R Console
> #入力ファイルの読み込み
> data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #確認$
> head(data) #確認$
      Length Count
NM_203348.1    3543     3
NM_001008737.1  1897    19
NM_001037228.1   537     7
NM_033183.2     886     0
NM_138368.3    4443    56
NM_152833.2    2844    85
> sum(data[,2]) #総リード$
[1] 2385273
>
> |
```



配列長 (RPK) 補正で...

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときの Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下を

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=FALSE)
```

①この2つの遺伝子発現レベルの大小関係に着目。生のカウント数だと19 vs. 7で NM_001008737.1のほうが多い。しかしながら、配列長も $1,897 / 537 = 3.53$ 倍長い。従って、配列長補正もかかったらRPKM値では大小関係が逆転するはず。

```
R Console
> #入力ファイルの読み込み
> data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #確認$
> head(data) #確認$
      gene ID      Length Count
NM_203348.1      3543         3
NM_001008737.1  1897         19
NM_001037228.1   537          7
NM_033183.2       886          0
NM_138368.3      4443         56
NM_152833.2      2844         85
> sum(data[,2]) #総リード数
[1] 2385273
>
> |
```



最後までコピー

コード全部を実行した結果のR Console画面は、こんな感じになります。①がRPKM値です。

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

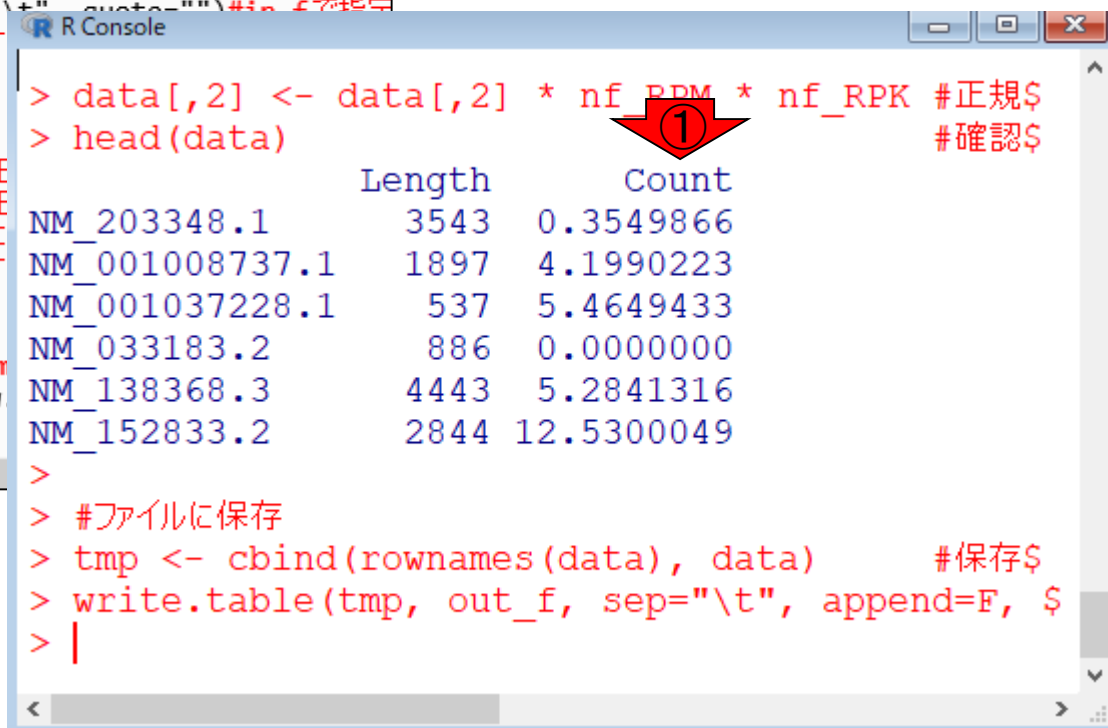
```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmp
write.table(tmp, out_f, sep="\t", append=F, row.names=F)
```



```
> data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化$
> head(data) #確認$
```

	Length	Count
NM_203348.1	3543	0.3549866
NM_001008737.1	1897	4.1990223
NM_001037228.1	537	5.4649433
NM_033183.2	886	0.0000000
NM_138368.3	4443	5.2841316
NM_152833.2	2844	12.5300049

```
>
> #ファイルに保存
> tmp <- cbind(row.names(data), data) #保存$
> write.table(tmp, out_f, sep="\t", append=F, $
> |
```


Raw count vs. RPKM

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったとき Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(rownames(data), data) #保存したい情報をtmp
write.table(tmp, out_f, sep="\t", append=F, row
```

①この2つの遺伝子発現レベルの大小関係に着目。生のカウント数だと19 vs. 7で NM_001008737.1のほうが多いが、配列長補正によりRPKM値では大小関係が逆転した。このように同一サンプル内での異なる遺伝子(feature)間の発現レベルの大小関係を知りたい場合は、配列長補正は必須です。

```
R Console
> data[,2] <- data[,2] * nf_RPM * nf_RPK #正規$
> head(data) #確認$
```

	Length	Count
NM_203348.1	3543	0.3549866
NM_001008737.1	1897	4.1990223
NM_001037228.1	537	5.4649433
NM_033183.2	886	0.0000000
NM_138368.3	4443	5.2841316
NM_152833.2	2844	12.5300049

```
>
> #ファイルに保存
> tmp <- cbind(rownames(data), data) #保存$
> write.table(tmp, out_f, sep="\t", append=F, $
> |
```

ちなみに...

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

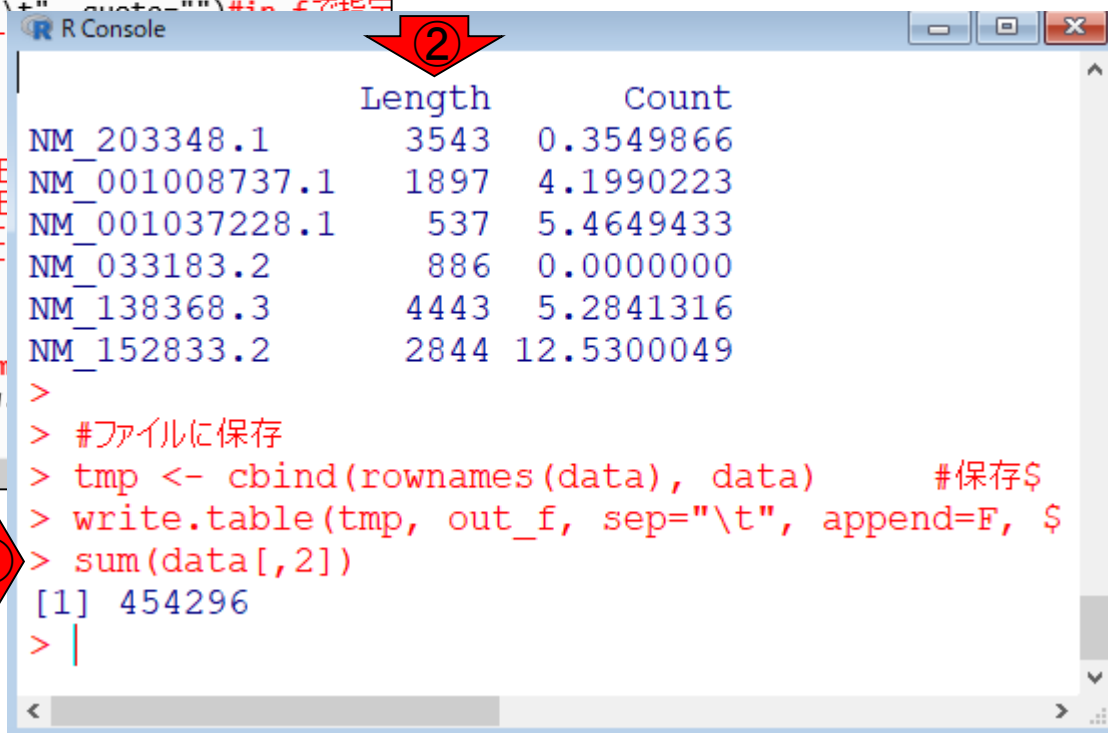
#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmp
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)
```

①RPKM値の総和は、通常100万ぴったりにはなりません。理由は配列長補正がかかったデータだからです。約45万という結果から、平均の配列長が2,000より長かったのだらうと予想できます。



```
R Console
> sum(data[,2])
[1] 454296
> |
> #ファイルに保存
> tmp <- cbind(row.names(data), data) #保存$
> write.table(tmp, out_f, sep="\t", append=F, $
> sum(data[,2])
[1] 454296
> |
```

	Length	Count
NM_203348.1	3543	0.3549866
NM_001008737.1	1897	4.1990223
NM_001037228.1	537	5.4649433
NM_033183.2	886	0.0000000
NM_138368.3	4443	5.2841316
NM_152833.2	2844	12.5300049

①

②

ちなみに...

正規化 | 基礎 | RPKM

遺伝子(転写物)ごとのリード数を「配列長が1000 bp (kilobase)で総リード数が100万だったときのリード数; Reads per kilobase per million (RPKM)」に変換するやり方を示します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. 配列長とカウント情報を含むファイル(sample_length_count.txt)の場合:

1-3列目がそれぞれ、gene ID, 配列長, カウント数からなるファイルです。

```
in_f <- "sample_length_count.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定
head(data) #確認してるだけです
sum(data[,2]) #総リード数を表示
```

#本番(正規化)

```
nf_RPM <- 1000000/sum(data[,2]) #正規化係数(RPM補正)
nf_RPK <- 1000/data[,1] #正規化係数(RPK補正)
data[,2] <- data[,2] * nf_RPM * nf_RPK #正規化係数を各行に
head(data) #確認してるだけです
```

#ファイルに保存

```
tmp <- cbind(row.names(data), data) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=FALSE)
```

①配列長の平均値を計算。予想通りですね。
 $1,000,000 / 2,668.197 = 374.8$ でないからオカシイと思われるかもしれませんが、これは遺伝子ごとにカウント数が異なるためです。

```
R Console
NM_001008737.1 1897 4.1990223
NM_001037228.1 537 5.4649433
NM_033183.2 886 0.0000000
NM_138368.3 4443 5.2841316
NM_152833.2 2844 12.5300049
>
> #ファイルに保存
> tmp <- cbind(row.names(data), data) #保存$
> write.table(tmp, out_f, sep="\t", append=F, $
> sum(data[,2])
[1] 454296
> mean(data[,1])
[1] 2668.197
> |
```

