

Introduction to analyzing NanoString nCounter data using the NanoStringNormCNV package

Dorota Sendorek

September 29, 2017

Contents

1	Getting started	2
2	Setting Up Data	2
3	Quality Control Metrics	3
3.1	Positive controls	3
3.2	Restriction fragmentation controls	3
3.3	Invariant controls	4
4	Normalization Methods	5
4.1	Code Count Correction	6
4.2	Background Correction	6
4.3	Sample Content Correction	6
4.4	Other Methods	7
4.5	Wrapper Functions	7
4.6	Collapsing Probes	9
5	Calling CNAs	9
6	Evaluating Results	12
7	Visualization	12

1 Getting started

NanoStringNormCNV is a suite of tools used to perform quality control, pre-processing, copy number calling and visualization on NanoString nCounter DNA data. NanoString is a medium-throughput platform which first gained popularity through gene expression quantification and now has extended its functionality to copy number variation (CNV) detection in genomic DNA. NanoString holds a number of advantages over traditional assays and these include its ability to handle lower quality samples, measure DNA without necessary amplification, and produce an absolute nucleic acid count [1]. Given this, the applications of this technology are wide-ranging, from validation to clinical.

NanoStringNormCNV has been created to aid users in identifying optimal CNV data processing techniques for their datasets, as well as providing a data analysis foundation for the community to build on. The package both implements NanoString-recommended data processing instructions [2] and includes additional processing options and features. NanoStringNormCNV extends the NanoStringNorm package [3], enabling the utilization and expansion of those pre-processing techniques too.

This vignette details the workflow of NanoStringNormCNV. Basic steps include loading the data, running quality control metrics, pre-processing the raw data, calling CNVs, and evaluating and visualizing results. Small example datasets are provided for learning purposes.

2 Setting Up Data

Raw NanoString data is often provided in RCC file format. These files can be imported using `NanoStringNorm::read.markup.RCC` or `NanoStringNorm::read.xls.RCC`.

Alternatively, data can be read in from a raw text file as a data frame and manually formatted to fit NanoStringNormCNV specifications. The user must ensure that the first three columns of the data frame are 'CodeClass' (probe type), 'Name' (unique probe name), and 'Accession' (name of corresponding gene or genomic segment). The following columns must hold raw sample counts with sample IDs for column names. An example dataset for NanoString raw counts is provided here.

```
> require('NanoStringNormCNV');
> # load raw count example dataset
> data("NanoString");
> print(NanoString[1:6, 1:7]);
```

	CodeClass	Name	Accession	CPCG0103P3	CPCG0103P5	CPCG0103P7	CPCG0183F1.M1
2	Endogenous	TP73-1	TP73	1054	197	1004	906
3	Endogenous	TP73-2	TP73	182	68	287	304
4	Endogenous	TP73-3	TP73	591	147	670	776
5	Endogenous	MYCL1-1	MYCL1	548	233	1120	882
6	Endogenous	MYCL1-2	MYCL1	278	104	595	571
7	Endogenous	MYCL1-3	MYCL1	180	64	290	315

The sample annotation file (csv or tab-delimited) is loaded separately. It is recommended to use `load.phenodata` as this function checks that the information and formatting meets package requirements for downstream processing. See "PhenoData" man page for details on formatting requirements.

```
> # load annotation example dataset
> data("PhenoData");
> # optionally, read in annotation file (same information as above)
> PhenoData <- load.phenodata(
+   fname = system.file("extdata", "PhenoData.tsv", package = "NanoStringNormCNV"),
+   separator = "tab"
+ );
> print(head(PhenoData));
```

	SampleID	Patient	Name	Cartridge	Type	ReferenceID	HasReplicate	Sex	Fragmentation
1	CPCG0103P3	CPCG0103	CPCG0103.P3	1	Tumour	missing	0	M	Sonicate
2	CPCG0103P5	CPCG0103	CPCG0103.P5	1	Tumour	missing	0	M	Sonicate
3	CPCG0103P7	CPCG0103	CPCG0103.P7	1	Tumour	missing	0	M	Sonicate
4	CPCG0183F1.M1	CPCG0183	CPCG0183.F1	2	Tumour	missing	1	M	Sonicate
5	CPCG0183F1.M2	CPCG0183	CPCG0183.F1	2	Tumour	missing	1	M	AluI
6	CPCG0184F1.M1	CPCG0184	CPCG0184.F1	1	Tumour	missing	1	M	Sonicate

3 Quality Control Metrics

There are three quality controls metrics available, each utilizing one of the three control probe types included in NanoString CNV CodeSets.

3.1 Positive controls

Positive control probe counts are correlated with expected concentrations. In accordance with NanoString guidelines [2], raw counts (x) are first converted to target concentrations (y) using the following equation:

$$y = 171.23x + 214.12 \quad (1)$$

Samples with R squared values that fall below 0.95 are flagged. Additionally, diagnostic plots can be generated to visualize results.

```
> # quality control using positive controls
> r.squared <- positive.control.qc(raw.data = NanoString);
> # plot R squared values
> make.positive.control.plot(correlations = r.squared, covs = PhenoData);
```

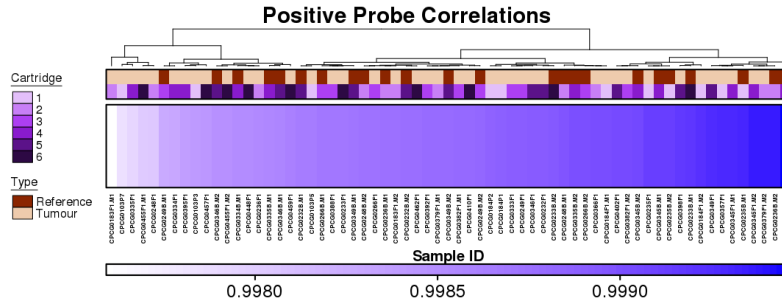


Figure 1: R squared values across all samples. All samples are well above the 0.95 cutoff.

3.2 Restriction fragmentation controls

Restriction fragmentation controls are used to determine whether DNA denaturation and digestion has occurred [2]. This metric applies only to AluI-digested samples. Probes that contain AluI restriction sites ($A+B$) are compared to probes that do not ($C+D$). Complete denaturation is signified by mean $C+D$ probe counts of > 200 . Restriction enzyme digestion is considered complete if $\frac{C+D}{A+B} \geq 10$. Samples with low counts or low ratios are flagged and visualized.

```
> # correctly running QC on AluI-digested samples only
> excl.samples <- PhenoData$SampleID[PhenoData$Fragmentation != "AluI"];
> probe.ratios <- restriction.fragmentation.qc(
+   raw.data = NanoString[, ! names(NanoString) %in% excl.samples]
+ );
```

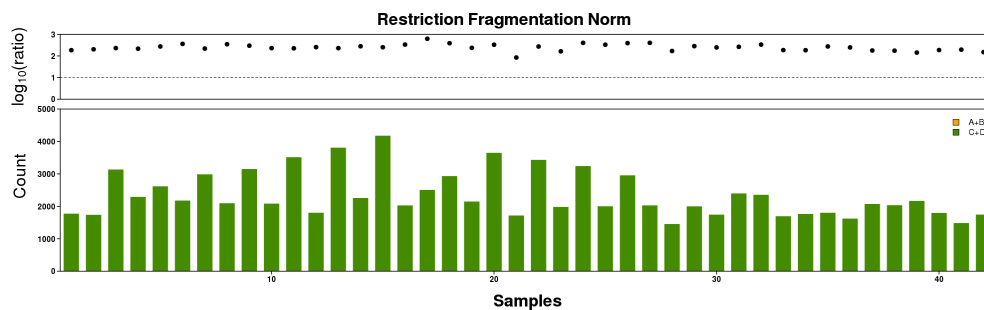


Figure 2: Top plot displays restriction fragmentation probe ratios. All sample ratios are well above the minimum of 10 (denoted by the dashed line). The bottom plot displays raw restriction fragmentation probe counts. All samples have $C + D$ counts of over 200.

```
> # running QC on all available samples (\textit{i.e.} AluI-digested and sonicated)
> probe.ratios <- restriction.fragmentation.qc(
+   raw.data = NanoString
+ );
```

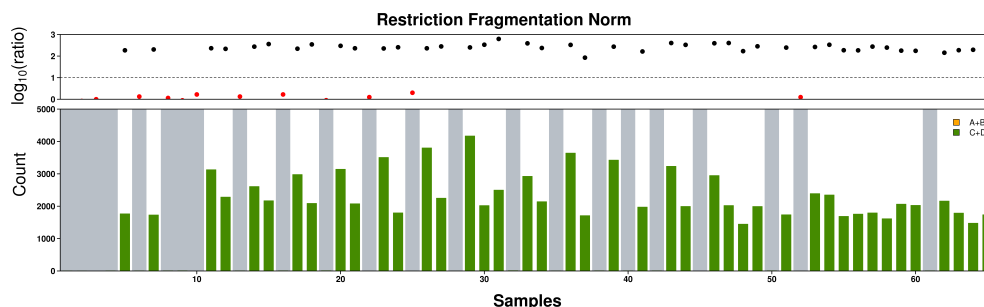


Figure 3: Samples that did not undergo enzyme digestion fail this step. DNA from samples fragmented by sonication (as opposed to AluI digestion) have small $\frac{C+D}{A+B}$ probe ratios and can be identified by red dots in the top plot and grey highlighting in the bottom plot.

3.3 Invariant controls

The third quality control step involves plotting invariant control probe counts. Samples found to have mean invariant counts of less than 100 are considered low quality [2]. Low counts in normal reference samples are especially problematic as they lead to exaggerated copy number calls.

```
> # plotting invariant probes
> make.invariant.probe.plot(
+   inv.probe.counts = NanoString[NanoString$CodeClass == 'Invariant', -(1:3)],
+   tissue.type = PhenoData
+ );
```

If there are low count invariant probes, two diagnostic plots are produced. One of these is a barplot showing samples with invariant probe counts of less than 100.

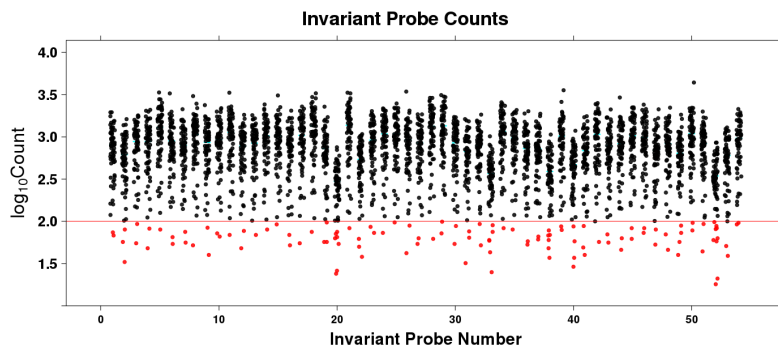


Figure 4: Invariant counts per probe across all samples. Counts that fall under the recommended minimum of 100 are in red.

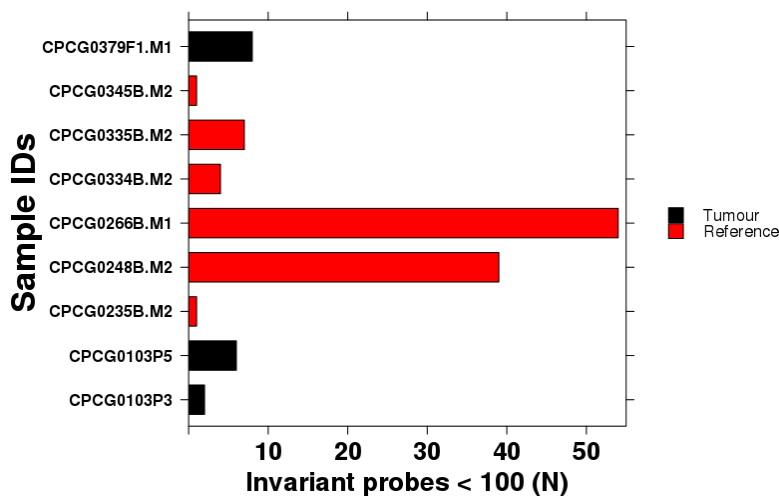


Figure 5: The number of probes with low counts per sample. Normal reference sample bars are in red and tumour sample bars are in black. Low counts are most problematic in reference samples when calling copy number downstream.

In this example, samples CPCG0266B.M1 and CPCG0248B.M2 are found to have unusually large numbers of low count probes. As such, here we remove these poor quality reference samples from both the raw count and annotation datasets.

```
> low.quality <- c('CPCG0266B.M1', 'CPCG0248B.M2');
> NanoString <- NanoString[, !names(NanoString) %in% low.quality];
> PhenoData <- PhenoData[!PhenoData$SampleID %in% low.quality,];
```

It is important to remember that the annotation data must be updated to reflect these changes prior to continuing with data processing. As CPCG0248B.M2 was the matched normal for tumour sample CPCG0248F1, the reference ID for CPCG0248F1 must be set to 'missing'. Additionally, since both removed samples had replicates, the 'HasReplicate' column must also be updated.

```
> # update matched normal and replicate information, as necessary
> PhenoData[PhenoData$SampleID == 'CPCG0248F1',]$ReferenceID <- 'missing';
> PhenoData[PhenoData$SampleID %in% c('CPCG0266B.M2', 'CPCG0248B.M1'),]$HasReplicate <- 0;
```

4 Normalization Methods

Most of the normalization options available in NanoStringNormCNV are accessed through NanoStringNorm. The options are briefly outlined below but for full details please see the NanoStringNorm vignette 'Introduction to analyzing NanoString nCounter data using the NanoStringNorm package' [3].

4.1 Code Count Correction

The first option is code count correction ('cc'). Positive controls (PC) are used to minimize any lane-to-lane variation. A normalization factor (NF) is calculated from the PC counts by summarizing the counts using summation ('sum') or by taking the geometric mean ('geo.mean'). All raw counts are then multiplied by the NF. To skip this step, specify 'none'.

Code count correction using summation for probe summarization:

$$PC_s = \sum_{g=1}^{n_G} PC_{g,s}$$

$$NF_s = \frac{\frac{1}{n_S} \times \sum_{s=1}^{n_S} PC_s}{PC_s} \quad (2)$$

$$x_{adj} = x_{unadj} \times NF_s$$

Where x = data matrix[g,s], s = sample index, g = gene index, n_S = number of samples and n_G = number of genes.

4.2 Background Correction

The second option is background correction ('bc'). Negative controls (NC) are used to account for non-specific binding to probes. To calculate background noise, NC are summarized either by taking the mean ('mean'), the maximum ('max') or the mean plus two standard deviations ('mean.2sd'). The background is then subtracted from each sample. To skip, specify 'none'.

Background correction using mean plus 2 standard deviations for probe summarization:

$$\overline{NC}_s = \frac{1}{n_G} \times \sum_{g=1}^{n_G} NC_{g,s}$$

$$\sigma_{NC_s} = \sqrt{\frac{1}{n_G} \times \sum_{g=1}^{n_G} (x_{g,s} - \overline{NC}_s)^2} \quad (3)$$

$$NC_s = \overline{NC}_s + 2 \times \sigma_{NC_s}$$

$$x_{adj} = x_{unadj} \times NC_s$$

Where \overline{NC} = mean of negative control counts, σ_{NC} = standard deviation of negative control counts, x = data matrix[g,s], s = sample index, g = gene index, n_S = number of samples and n_G = number of genes.

4.3 Sample Content Correction

The third option is sample content correction ('sc'). Housekeeping genes (HK) are used to normalize for genomic DNA input amounts (*e.g.* differences due to pipetting fluctuations). To normalize, HK counts are summarized to calculate a normalization factor (NF) by which all counts are multiplied. A number of options are available for probe summarization: 'housekeeping.geo.mean', 'total.sum', 'top.geo.mean', 'low.cv.geo.mean'. To skip, specify 'none'.

For 'housekeeping.geo.mean', NanoStringNorm requires probes of code class 'Housekeeping' and will throw out an error if they are not provided [3]. Please note that in the dataset we provide, the 'Housekeeping' probes are either a) 'Endogenous' probes that were found to show little variability in counts across data or b) simulated counts

created by adding noise to existing housekeeping genes (these are denoted by the name prefix 'SIM'). The code class of these genes was set to 'Housekeeping' manually.

Method 'top.geo.mean' uses the geometric mean of the top 75 expressed genes whereas 'low.cv.geo.mean' uses the geometric mean of the genes with the lowest coefficients of variation (recommended if outliers are present). Finally, 'total.sum' uses the sum of all the probes.

Sample content correction using geometric mean for probe summarization:

$$HK_s = \left(\sqrt[n_G]{\prod HK_g} \right)_s$$

$$NF_s = \frac{\frac{1}{n_S} \times \sum_{s=1}^{n_S} HK_s}{HK_s} \quad (4)$$

$$x_{adj} = x_{unadj} \times NF_s$$

Where x = data matrix[g,s], s = sample index, g = gene index, n_S = number of samples and n_G = number of genes.

A new option for sample content correction (not implemented in NanoStringNorm) makes use of the invariant control (IC) probes. This method can be run directly with `invariant.probe.norm` or in combination with the other normalization techniques. Invariant probe normalization is the method outlined in NanoString guidelines for CNV analysis [2] and therefore recommended here for sample content correction.

Invariant probe normalization:

$$\overline{IC}_s = \frac{1}{n_G} \times \sum_{g=1}^{n_G} IC_{g,s}$$

$$NF_s = \frac{\frac{1}{n_S} \times \sum_{s=1}^{n_S} \overline{IC}_s}{\overline{IC}_s} \quad (5)$$

$$x_{adj} = x_{unadj} \times NF_s$$

Where \overline{IC} = mean of invariant control counts, x = data matrix[g,s], s = sample index, g = gene index, n_S = number of samples, n_G = number of genes and NF = normalization factor.

Ultimately, invariant and/or housekeeping probes are selected so as to have negligible variability in probe count across samples and experiments. Practically speaking, this is not always the case and excessive variability in these probes should be checked. Since CNA calling is performed on these probe types, users may use the results to identify probes with extreme values as an additional quality-control step.

4.4 Other Methods

The final option is for additional normalization methods, alternative to those proposed by NanoString, such as 'vsn' and 'quantile'. The 'quantile' method performs the following: it ranks gene counts per sample, calculates the median count per rank across samples and creates an empirical distribution of these median counts, to which each sample is transformed. For 'vsn', variance stabilizing normalization is applied via the R package vsn [4]. To skip this option, specify 'none'.

4.5 Wrapper Functions

Finally, to maximize flexibility, NanoString cartridges can either be processed independently or combined (recommended). If processed independently, normalization techniques are applied to samples from each cartridge

separately (ignoring probe information from other samples). If processed combined, all samples in the cohort are normalized together as a single batch. To accomplish this, we provide two wrapper functions: `normalize.global` and `normalize.per.chip`.

```
> # example 1
> # perform invariant probe normalization only --cartridges combined
> NanoString.norm <- normalize.global(
+   raw.data = NanoString,
+   cc = 'none',
+   bc = 'none',
+   sc = 'none',
+   oth = 'none',
+   do.rcc.inv = TRUE,
+   covs = NA,
+   phenodata = PhenoData
+ );

> # example 2
> # perform invariant probe normalization only --cartridges individually
> NanoString.norm <- normalize.per.chip(
+   raw.data = NanoString,
+   cc = 'none',
+   bc = 'none',
+   sc = 'none',
+   oth = 'none',
+   do.rcc.inv = TRUE,
+   covs = NA,
+   phenodata = PhenoData
+ );

INFO [2017-09-29 12:48:12] Normalizing cartridge 1
INFO [2017-09-29 12:48:18] Normalizing cartridge 2
INFO [2017-09-29 12:48:24] Normalizing cartridge 5
INFO [2017-09-29 12:48:28] All invariant probe counts pass minimum threshold of 100
INFO [2017-09-29 12:48:28] Normalizing cartridge 6
INFO [2017-09-29 12:48:31] All invariant probe counts pass minimum threshold of 100
INFO [2017-09-29 12:48:31] Normalizing cartridge 3
INFO [2017-09-29 12:48:34] All invariant probe counts pass minimum threshold of 100
INFO [2017-09-29 12:48:34] Normalizing cartridge 4

> # example 3
> # include covariates for sample cartridge and sample type
> # covariates must be binary as they are passed directly to NanoStringNorm 'traits'
> covs <- as.data.frame(matrix(
+   1,
+   nrow = nrow(PhenoData),
+   ncol = length(unique(PhenoData$Cartridge)),
+   dimnames = list(
+     PhenoData$SampleID,
+     paste0("Cartridge", unique(PhenoData$Cartridge))
+   )
+ ));
> for (n in 1:nrow(PhenoData)) {
+   covs[n, which(unique(PhenoData$Cartridge) == PhenoData$Cartridge[n])] <- 2;
+ }
> covs$Type <- ifelse(PhenoData$Type == 'Reference', 1, 2);
> NanoString.norm <- normalize.global(
+   raw.data = NanoString,
```



```

+         cc = 'none',
+         bc = 'none',
+         sc = 'none',
+         oth = 'none',
+         do.rcc.inv = TRUE,
+         covs = covs,
+         phenodata = PhenoData
+     );

> # same as above but per chip
> NanoString.norm <- normalize.per.chip(
+     raw.data = NanoString,
+     cc = 'none',
+     bc = 'none',
+     sc = 'none',
+     oth = 'none',
+     do.rcc.inv = TRUE,
+     covs = covs,
+     phenodata = PhenoData
+ );

INFO [2017-09-29 12:48:48] Normalizing cartridge 1
INFO [2017-09-29 12:48:54] Normalizing cartridge 2
INFO [2017-09-29 12:49:00] Normalizing cartridge 5
INFO [2017-09-29 12:49:04] All invariant probe counts pass minimum threshold of 100
INFO [2017-09-29 12:49:04] Normalizing cartridge 6
INFO [2017-09-29 12:49:07] All invariant probe counts pass minimum threshold of 100
INFO [2017-09-29 12:49:07] Normalizing cartridge 3
INFO [2017-09-29 12:49:11] All invariant probe counts pass minimum threshold of 100
INFO [2017-09-29 12:49:11] Normalizing cartridge 4

```

4.6 Collapsing Probes

Post-normalization, for datasets where there are multiple probes per gene/genomic segment, one may choose to collapse probes to a single value. Probes with matching 'Accession' values are collapsed by taking the mean of their counts. Unique probe 'Name' values are substituted with 'Accession' values.

```

> NanoString.norm.col <- collapse.genes(normalized.data = NanoString.norm);
> print(NanoString.norm.col[1:6, 1:6]);

```

	CodeClass	Name	Accession	CPCG0103P3	CPCG0103P5	CPCG0103P7
AKT2	Endogenous	AKT2	AKT2	721	732	740
AKT3	Endogenous	AKT3	AKT3	1054	816	742
APC	Endogenous	APC	APC	992	788	605
AR	Endogenous	AR	AR	344	386	393
AURKA	Endogenous	AURKA	AURKA	520	555	552
BBC3	Endogenous	BBC3	BBC3	987	680	642

5 Calling CNAs

A copy number aberration (CNA) is defined as a somatic (*de novo*) CNV. Two functions are provided to perform CNA calling on normalized data: `call.cnas.with.matched.normals` and `call.cnas.with.pooled.normals`. Additionally, calling can be performed on sex chromosome segments provided that sample sex information is provided in the annotation.

For each probe ('Endogenous', 'Invariant', 'Housekeeping'), tumour-normal ratios are calculated. If calling using matched normals, each tumour sample probe is divided by its corresponding matched normal (reference) probe.

$$TN_{g,s} = \frac{T_{g,s}}{R_{g,s}} \quad (6)$$

Where TN = tumour-normal ratio, T = tumour count data matrix[g,s], R = reference count data matrix[g,s], s = sample index and g = gene index.

If calling using pooled normals, each tumour sample probe is divided by the normal cohort's mean count of the corresponding probe.

$$TN_{g,s} = \frac{T_{g,s}}{\bar{R}_g} \quad (7)$$

Where TN = tumour-normal ratio, T = tumour count data matrix[g,s], \bar{R} = mean reference count, s = sample index and g = gene index.

Additionally for pooled normals, ratios for normal sample probes are also calculated.

$$NN_{g,s} = \frac{R_{g,s}}{\bar{R}_g} \quad (8)$$

Where NN = normal-normal_{mean} ratio, R = reference count data matrix[g,s], \bar{R} = mean reference count, s = sample index and g = gene index.

As suggested by NanoString guidelines [2], tumour-normal ratios are then boosted by a multiplication factor. For autosomal chromosome probe ratios, this is a multiplication factor of two, representing the diploid nature of the genome. This also applies to chromosome X probes in female samples. For sex chromosome probes in male samples, a multiplication factor of one is used instead. Furthermore, when calling copy number with pooled normals, ratios are additionally adjusted so that median sample copy number is equal to the multiplication factor.

$$X_{boost} = X_{unboost} \times MF \quad (9)$$

$$X_{adj} = X_{unadj} - (\tilde{X}_{unadj} - MF)$$

Where X = sample tumour-normal ratios, \tilde{X} = median sample tumour-normal ratio and MF = multiplication factor.

Next, ratios are converted to categorical CNA calls (0 to 4) by rounding. Rounding is performed using four thresholds representing 1) a homozygous deletion, 2) a heterozygous deletion, 3) a single-copy gain and 4) a multi-copy gain. Currently, there are three options available for setting the CNA thresholds:

1. Derived from NanoString manual recommendations [2]. Thresholds are: 0.4, 1.5, 2.5, 3.5
2. Calculated from normal sample cohort ratios. Note, this option is only available for `call.cnas.pooled.normals`. Sex chromosome probe information from male samples is excluded from the following calculations. The minimum and maximum ratios are obtained from each normal sample and the median minimum (min) and median maximum (max) ratios are selected. Using all normal sample ratios, standard deviation (SD) is calculated. Thresholds are: min, min + SD, max - SD, max
3. By applying the kernel-density approach. Thresholds for determining changes in copy number are derived from various quantiles of the kernel density of the tumour-normal ratios. We provide default cut point values for symmetric distributions: 0.85 (which translates to 7.5th quantile for -1 and 92.5th quantile for +1), 0.95 (for copy number changes of ± 2). However, the user is highly encouraged to identify cut points applicable to their dataset. For skewed distributions, users can supply four unique cut points, one each for copy number changes -2, -1, +1, +2. Note that kernel densities are sample-specific if using `call.cnas.with.matched.normals` and cohort-specific if using `call.cnas.with.pooled.normals`.

```
> # Option 1: call using matched normal reference
> cnas <- call.cnas.with.matched.normals(
+   normalized.data = NanoString.norm,
+   phenodata = PhenoData,
+   per.chip = FALSE,
+   call.method = 2,
+   kd.values = c(0.99, 0.87, 0.89, 0.96),
+   use.sex.info = TRUE
+ );
```

INFO [2017-09-29 12:49:18] Identified the following as sex chromosome probes:

```
chrXp113
chrXp1121
chrXq131
chrXq2131
chrXq2132
chrYp1131
chrYp112-5
chrYp112-6
chrYq1121
chrYq11221
```

WARN [2017-09-29 12:49:18] Low chrX/chrY probe number! Consider using pooled normals when calling CNAs in r

```
> # Option 2: call using a pooled normals reference
> cnas <- call.cnas.with.pooled.normals(
+   normalized.data = NanoString.norm,
+   phenodata = PhenoData,
+   per.chip = FALSE,
+   call.method = 3,
+   use.sex.info = TRUE
+ );
```

INFO [2017-09-29 12:49:19] Identified the following as sex chromosome probes:

```
chrXp113
chrXp1121
chrXq131
chrXq2131
chrXq2132
chrYp1131
chrYp112-5
chrYp112-6
chrYq1121
chrYq11221
```

```
> # Option 3: call using a pooled normals reference
> cnas <- call.cnas.with.pooled.normals(
+   normalized.data = NanoString.norm,
+   phenodata = PhenoData,
+   per.chip = FALSE,
+   call.method = 1,
+   use.sex.info = TRUE
+ );
```

INFO [2017-09-29 12:49:19] Identified the following as sex chromosome probes:

```

chrXp113
chrXp1121
chrXq131
chrXq2131
chrXq2132
chrYp1131
chrYp112-5
chrYp112-6
chrYq1121
chrYq11221

```

6 Evaluating Results

To determine the optimal schema for processing a dataset, it is recommended to run and compare several (if not all) pre-processing parameter combinations. Several functions have been added to facilitate this comparison.

Information from replicate samples can be used to determine the concordance of categorical CNA calls or variance in the continuous tumour-normal ratio space.

```

> # if technical replicates are available
> evaluation <- evaluate.replicates(
+   phenodata = PhenoData,
+   normalized.data = NanoString.norm,
+   cna.rounded = cnas$rounded
+ );

```

Another option is to calculate adjusted Rand indexes (ARI) which measure to what extent data clusters according to some feature (*e.g.* tissue type). This can be performed on either continuous (*i.e.* normalized counts) or discrete data (*i.e.* copy number calls).

```

> # how well does the data cluster around the patients from which samples were obtained
> patient.ari <- get.ari(
+   data.to.cluster = evaluation$cna.calls,
+   feature = PhenoData[match(colnames(evaluation$cna.calls), PhenoData$SampleID),]$Patient,
+   is.discrete = TRUE
+ );

> # how much does the data cluster around the cartridges on which the samples were processed
> # log values, if appropriate
> if (all(unlist(NanoString.norm) >= 0)) {
+   count.data <- log10(NanoString.norm[, -c(1:3)] + 1);
+ } else {
+   count.data <- NanoString.norm[, -c(1:3)];
+ }
> cartridge.ari <- get.ari(
+   data.to.cluster = count.data,
+   feature = PhenoData$Cartridge[match(colnames(NanoString.norm[, -(1:3)]), PhenoData$SampleID)],
+   is.discrete = FALSE
+ );

```

7 Visualization

A number of plotting functions are provided. Diagnostic plots have been described previously (see section 'Quality Control Metrics'). Normalized and copy number data can also be visualized. The plotting wrapper function `visualize.results` will create figures using all available plotting functions. Visualization uses the R package BPG [5].

```

> # plot normalized NanoString counts
> make.counts.heatmap(
+     nano.counts = NanoString.norm[, -(1:3)],
+     fname.stem = 'normalized',
+     covs.rows = PhenoData[, c('SampleID', 'Type', 'Cartridge')],
+     covs.cols = NanoString[, c('Name', 'CodeClass')]
+ );

```

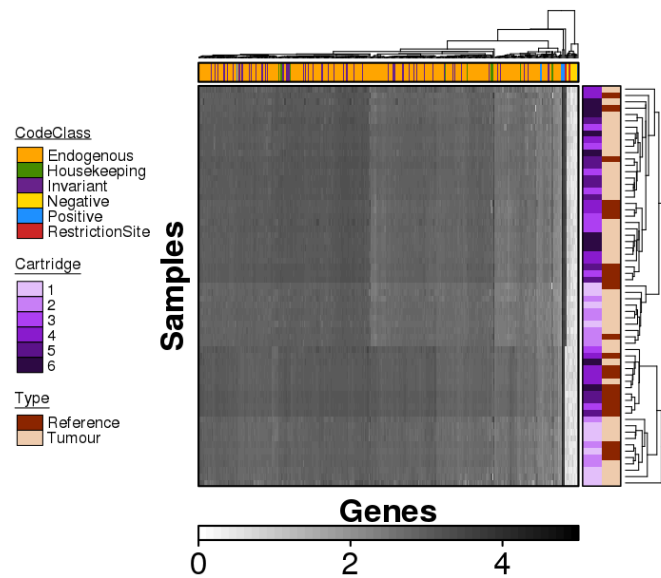


Figure 6: Normalized NanoString counts

```

> # plot raw NanoString counts
> # make sure raw count data frame has gene names for row names!
> NanoString.formatted <- NanoString[, -(1:3)];
> rownames(NanoString.formatted) <- NanoString$Name;
> make.counts.heatmap(
+     nano.counts = NanoString.formatted,
+     fname.stem = 'raw',
+     covs.rows = PhenoData[, c('SampleID', 'Type', 'Cartridge')],
+     covs.cols = NanoString[, c('Name', 'CodeClass')]
+ );

```

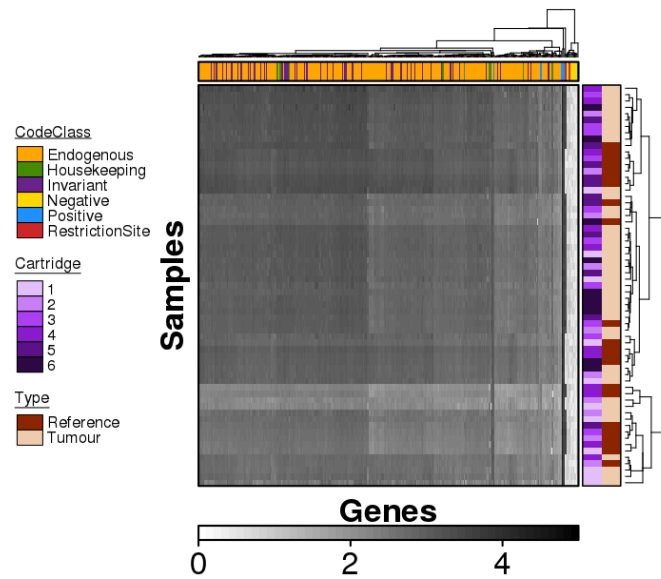


Figure 7: Raw NanoString counts

```

> # plot rounded copy number calls
> make.cna.heatmap(
+     nano.cnas = cnas$rounded,
+     fname.stem = 'round',
+     covs.rows = PhenoData[, c('SampleID', 'Type', 'Cartridge')],
+     covs.cols = NanoString[, c('Name', 'CodeClass')],
+     rounded = TRUE
+ );

```

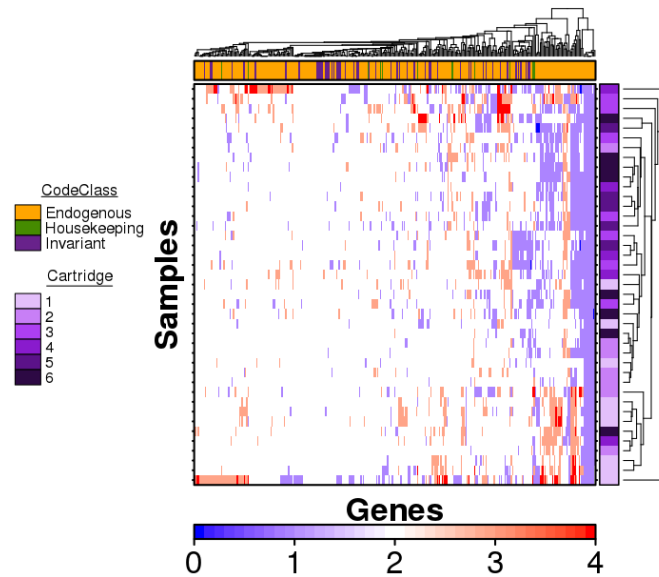


Figure 8: Rounded CNA calls

```

> # plot raw (not rounded) copy number calls
> # first, setting max copy number value at 5
> cnas.raw.max5 <- cnas$raw;
> cnas.raw.max5[cnas.raw.max5 > 5] <- 5;
> make.cna.heatmap(
+     nano.cnas = cnas.raw.max5,
+     fname.stem = 'raw',
+     covs.rows = PhenoData[, c('SampleID', 'Type', 'Cartridge')],
+     covs.cols = NanoString[, c('Name', 'CodeClass')],
+     rounded = FALSE
+ );

```

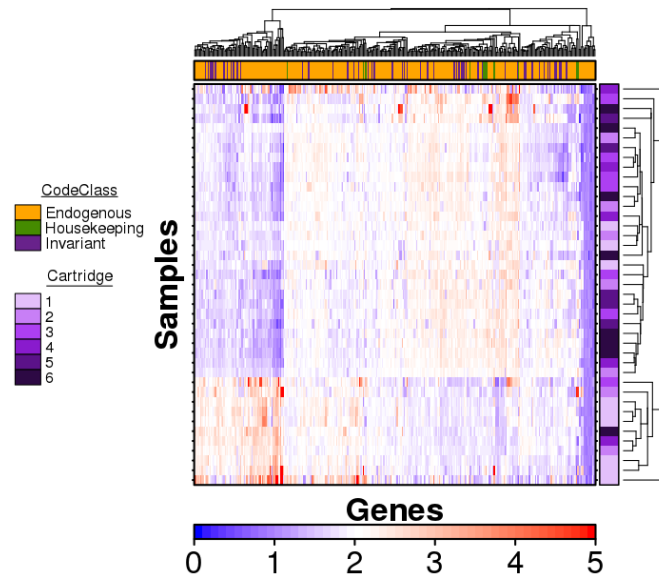


Figure 9: Raw CNA calls


```

> # plot copy number call density for rounded values
> # two plots: per gene and per sample
> make.cna.densities.plots(
+     nano.cnas = cnas$rounded
+ );

```

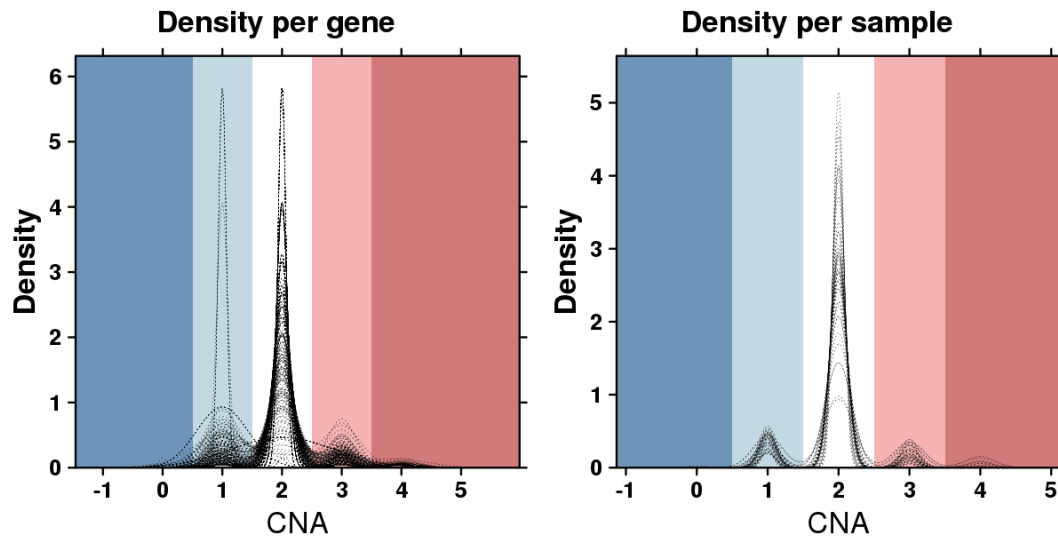


Figure 10: Rounded CNA call densities per gene (left) and per sample (right)

```

> # plot raw NanoString count correlations
> make.sample.correlations.heatmap(
+     nano.counts = NanoString.formatted,
+     covs = PhenoData[, c('SampleID', 'Cartridge', 'Type')]
+ );

```

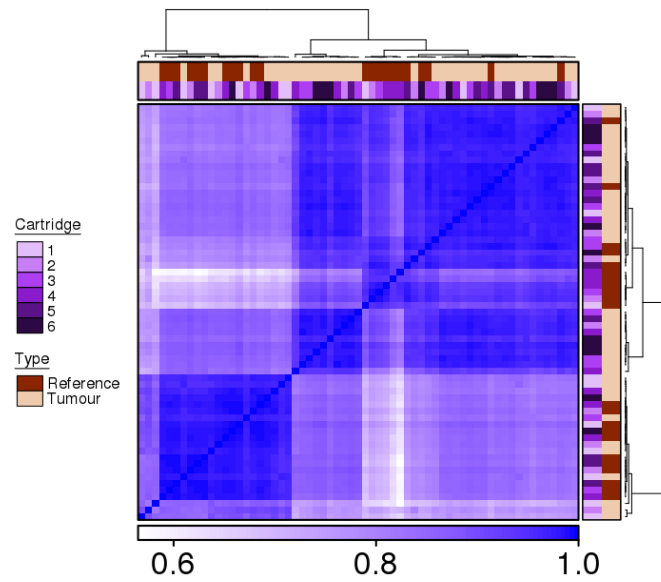


Figure 11: Raw NanoString count inter-sample correlations

```

> # alternatively, plot all results using wrapper function
> visualize.results(
+   raw.data = NanoString,
+   normalized.data = NanoString.norm,
+   phenodata = PhenoData,
+   cna.rounded = cnas$rounded,
+   cna.raw = cnas$raw,
+   replicate.eval = evaluation,
+   max.cn = 5
+ );

```

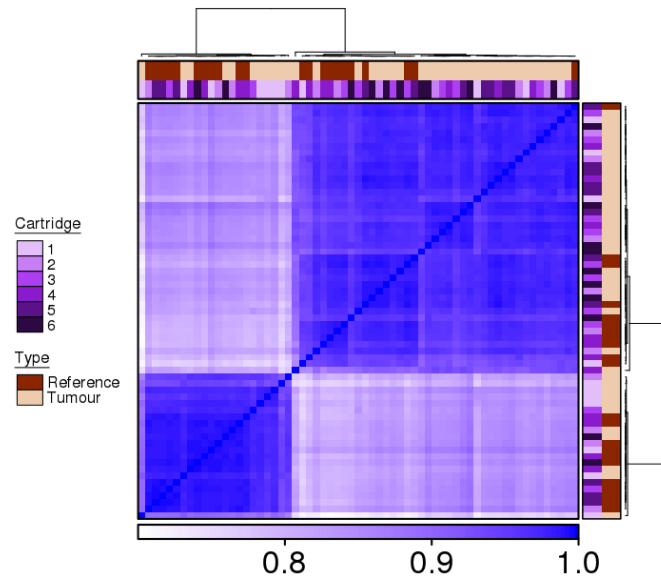


Figure 12: Normalized NanoString count inter-sample correlations

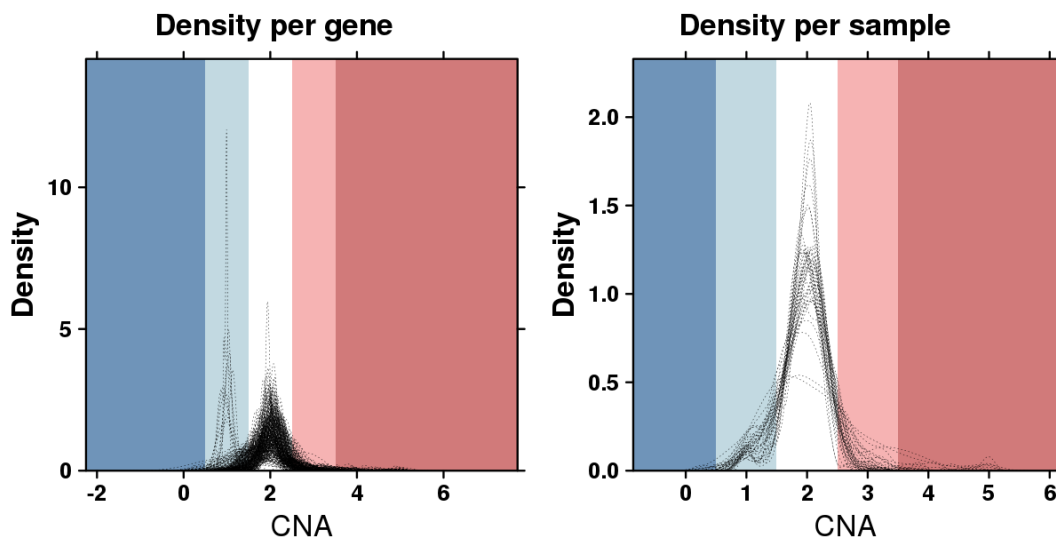


Figure 13: Raw CNA call densities per gene (left) and per sample (right)

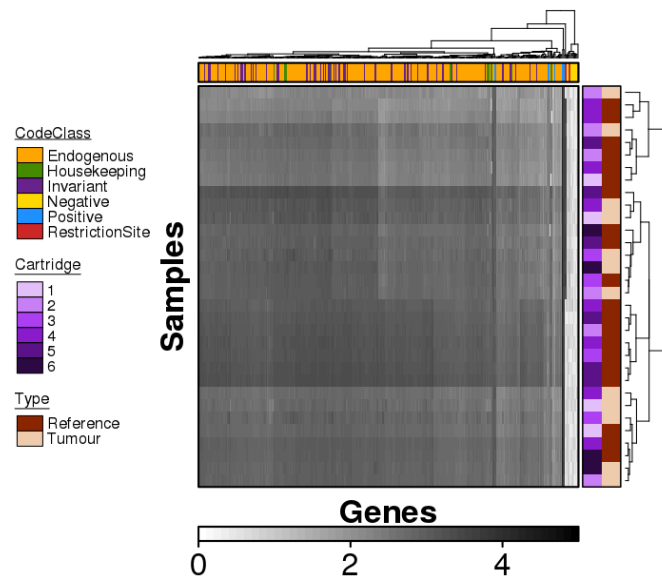


Figure 14: Raw counts for replicate samples

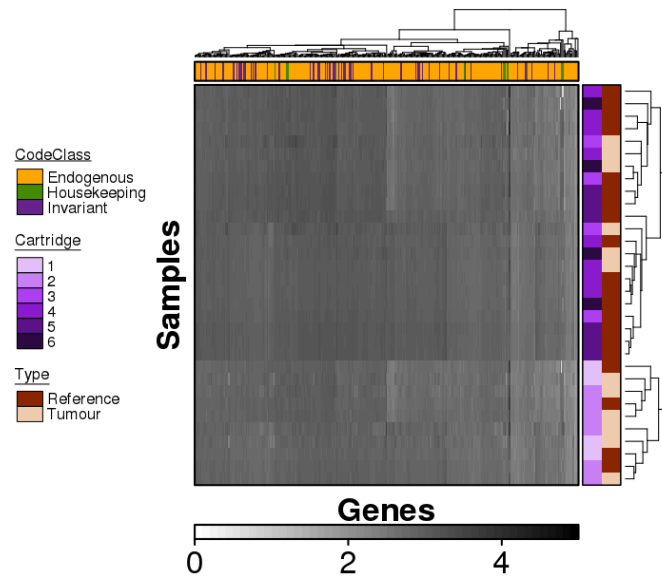


Figure 15: Normalized counts for replicate samples

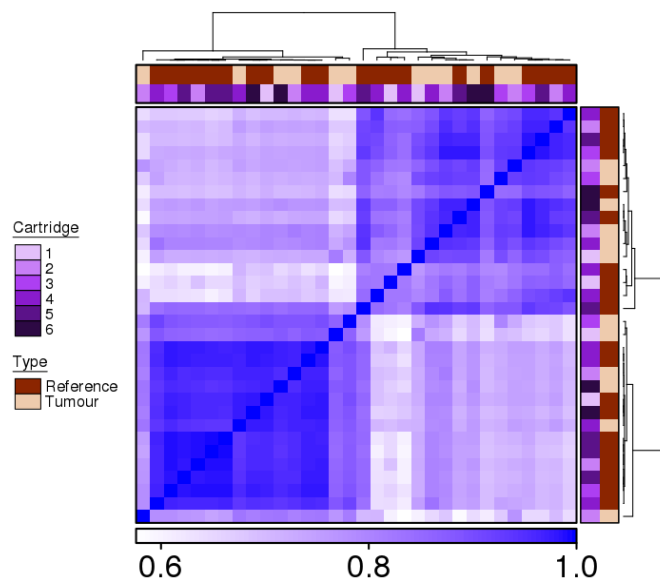


Figure 16: Normalized count inter-sample correlations for replicate samples

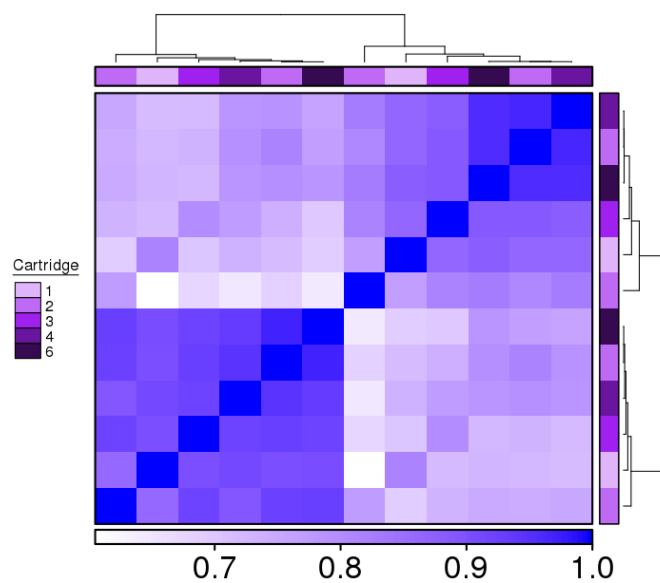


Figure 17: Normalized count inter-sample correlations for replicate samples, tumour only

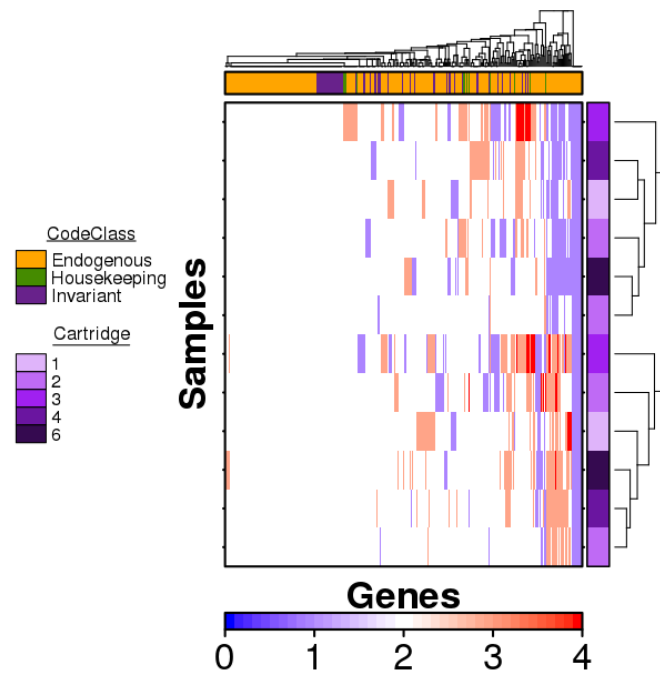


Figure 18: CNA calls for replicate samples

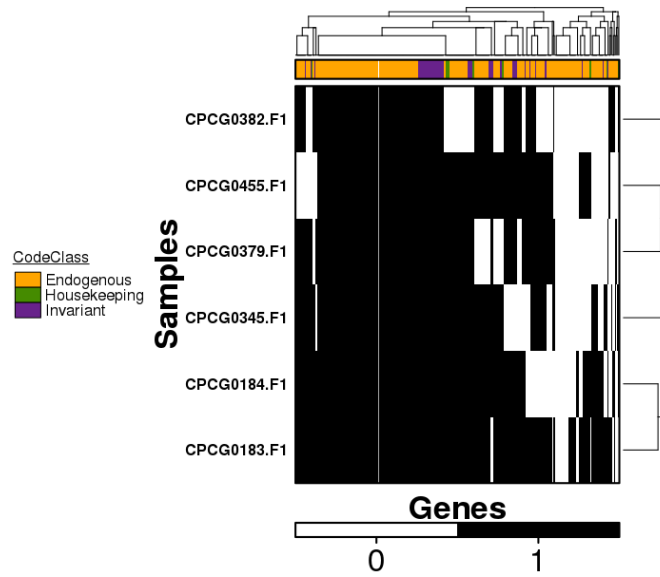


Figure 19: Copy number concordance across replicate samples

References

- [1] Geiss, G.K. *et al.* (2008) Direct multiplexed measurement of gene expression with color-coded probe pairs. *Nat Biotechnol*, 26(3):317-25. [2](#)
- [2] NanoString Technologies, Inc. (2011) nCounter®: Data Analysis Guidelines for Copy Number Variation (CNV). <<https://www.nanostring.com/support/product-support/support-documentation>> [2](#), [3](#), [4](#), [7](#), [10](#)
- [3] Waggott, D. *et al.* (2012) NanoStringNorm: an extensible R package for the pre-processing of NanoString mRNA and miRNA data. *Bioinformatics*, 28:1546-1548. [2](#), [5](#), [6](#)
- [4] Huber, W. *et al.* (2002) Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18:S96-S104. [7](#)
- [5] P'ng, C. *et al.* (2017) BPG: seamless, automated and interactive visualization of scientific data. *bioRxiv* 156067; doi: <https://doi.org/10.1101/156067>. [12](#)