Telomeric allelic imbalance reflects DNA repair defects and sensitivity to DNA damaging agents in cancer
Supplementary Methods - Data Analysis

Birkbak et al

February 6, 2012

This document describes the data analysis performed on all data.

The analysis was performed in the R statistical environment, and loads several objects containing pre-processed data. All these objects can be re-created from the data published with this manuscript, but can also be obtained from the authors on request.

All figures in this document is presented in their raw unpolished form, which occasionally means that the labeling may differ a bit from what is presented in the main manuscript, and be located at odd locations.

Contents

1 Getting started

1.1 Required Packages

```
> library(affy)
> library(squash)
> library(genefilter)
> library(beeswarm)
> library(gplots)
> library(gdata)
> library(ROC)
> library(DNAcopy)
> library(quantsmooth)
> library(cluster)
> library(jetset)
> library(boot)
```

1.2 A few functions

This function puts a label on the top left corner of a figure

```
> label.panel <- function(txt, xoff = 1, yoff = 2, cex = 1.2) {
+     x <- grconvertX(0, from = "nfc") + (xoff * strwidth("m"))
+     y <- grconvertY(1, from = "nfc") - (yoff * strheight("A"))
+     text(x, y, labels = txt, font = 2, xpd = TRUE, cex = cex)
+ }
```

This function re-organizes the output from running the ASCAT algorithm into a more compact format similar to the output from other segmentation algorithms such as Circular Binary Segmentation

```
> organize.ascat.segments <- function(ascat.output, markers) {
+     samp.names <- colnames(ascat.output$nA)
+     failed.names <- ascat.output$failedarrays
+     if (length(failed.names) >= 1) {
+         failed.locations <- which(is.na(ascat.output$segments))
+         new.samp.names <- 1:length(ascat.output$segments)
+         new.samp.names[-c(failed.locations)] <- samp.names
+         new.samp.names[failed.locations] <- failed.names
+         samp.names <- new.samp.names
+     }
+     out.seg <- matrix(0, 0, ncol = 7)
+     colnames(out.seg) <- c("SampleID", "Chr", "Start", "End",
+         "nProbes", "nA", "nB")
+     if (length(ascat.output$segments) != length(samp.names)) {
+         stop
+     }
+     for (i in 1:length(ascat.output$segments)) {
+         sample.segs <- ascat.output$segments[[i]]
+         if (class(sample.segs) != "matrix") {
+             next
+         }
+         if (class(sample.segs) == "matrix") {
+             for (j in 1:nrow(sample.segs)) {
+                 tmp <- markers[sample.segs[j, 1]:sample.segs[j,
```

```
+                   2], ]
+                 if (!all(tmp[, 1] == tmp[1, 1])) {
+                   stop("Error in segmentation")
+                 }
+                 chr <- as.character(tmp[1, 1])
+                 seg.start <- as.character(tmp[1, 2])
+                 seg.end <- as.character(tmp[nrow(tmp), 2])
+                 nProbes <- nrow(tmp)
+                 out.seg <- rbind(out.seg, c(samp.names[i], chr,
+                   seg.start, seg.end, nProbes, sample.segs[j,
+                     3:4]))
+             }
+             gc()
+         }
+     }
+     out.seg[out.seg[, 2] == "X", 2] <- 23
+     out.seg <- as.data.frame(out.seg)
+     out.seg[, 2:7] <- apply(out.seg[, 2:7], 2, function(x) {
+         as.numeric(as.character(x))
+     })
+     out.seg[, 1] <- as.character(out.seg[, 1])
+     return(out.seg)
+ }
```

This function defines and summarizes the regions of telomeric allelic imbalance

```
> no.tel <- function(seg.out, chrominfo, min.size = 0, min.probes = 1,
+     max.size = 1e+09, cnv.check = "no", cnv.seg = NULL, cnv.gain = NULL) {
+     if (class(seg.out) == "DNAcopy") {
+         seg.out <- seg.out$output
+     }
+     tmp.segs <- seg.out[!seg.out[, 2] %in% c("MT", "Y", "24"),
+         ]
+     tmp.segs <- tmp.segs[!tmp.segs[, 5] < min.probes, ]
+     tmp.segs[, 2] <- as.character(tmp.segs[, 2])
+     if (nrow(tmp.segs[tmp.segs[, 2] == "X", ]) > 0) {
+         tmp.segs[tmp.segs[, 2] == "X", 2] <- rep(23, nrow(tmp.segs[tmp.segs[,
+             2] == "X", ]))
+     }
+     if (cnv.check != "no") {
+         if (class(cnv.seg) == "DNAcopy") {
+             cnv.seg <- cnv.seg$output
+         }
+         tmp.cnv <- cnv.seg[!cnv.seg[, 2] %in% c("MT", "Y", "24"),
+             ]
+         tmp.cnv <- tmp.cnv[!tmp.cnv[, 5] < min.probes, ]
+         tmp.cnv[, 2] <- as.character(tmp.cnv[, 2])
+         if (nrow(tmp.cnv[tmp.cnv[, 2] == "X", ]) > 0) {
+             tmp.cnv[tmp.cnv[, 2] == "X", 2] <- rep(23, nrow(tmp.cnv[tmp.cnv[,
+                 2] == "X", ]))
+         }
+     }
+     tmp.segs[tmp.segs[, 6] == 1, 6] <- 2
+     for (j in 1:length(unique(tmp.segs[, 1]))) {
```

```
+         tmp.sample <- tmp.segs[tmp.segs[, 1] == unique(tmp.segs[,
+             1])[j], ]
+         for (i in 1:23) {
+             tmp1 <- tmp.sample[tmp.sample[, 2] == i, , drop = F]
+             if (tmp1[1, 6] == 2 & nrow(tmp1) != 1 & tmp1[1, 4] <
+                 (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 1
+                 if (cnv.check != "no") {
+                   if (cnv.seg[cnv.seg[, 1] == unique(tmp.segs[,
+                     1])[j] & cnv.seg[, 2] == i, ][1, 6] > cnv.gain) {
+                     tmp.sample[tmp.sample[, 2] == i, 6][1] <- 0
+                   }
+                 }
+             }
+             if (tmp1[nrow(tmp1), 6] == 2 & nrow(tmp1) != 1 &
+                 tmp1[nrow(tmp1), 3] > (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][nrow(tmp.sample[tmp.sample[,
+                   2] == i, ])] <- 1
+                 if (cnv.check != "no") {
+                   if (cnv.seg[cnv.seg[, 1] == unique(tmp.segs[,
+                     1])[j] & cnv.seg[, 2] == i, ][nrow(cnv.seg[cnv.seg[,
+                     1] == unique(tmp.segs[, 1])[j] & cnv.seg[,
+                     2] == i, ]), 6] > cnv.gain) {
+                     tmp.sample[tmp.sample[, 2] == i, 6][nrow(tmp.sample[tmp.sample[,
+                       2] == i, ])] <- 0
+                   }
+                 }
+             }
+             if (nrow(tmp.sample[tmp.sample[, 2] == i, ]) == 1 &
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] != 0) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 3
+             }
+         }
+         tmp.segs[tmp.segs[, 1] == unique(tmp.segs[, 1])[j], ] <- tmp.sample
+     }
+     no.events <- matrix(0, nrow = length(unique(tmp.segs[, 1])),
+         ncol = 28)
+     rownames(no.events) <- unique(tmp.segs[, 1])
+     colnames(no.events) <- c("Telomeric AI", "Mean size", "Interstitial AI",
+         "Mean Size", "Wholo chr AI", 1:23)
+     for (i in unique(tmp.segs[, 1])) {
+         tmp <- tmp.segs[tmp.segs[, 1] == i, ]
+         tmp <- tmp[(tmp[, 4] - tmp[, 3]) > min.size, ]
+         tmp <- tmp[(tmp[, 4] - tmp[, 3]) < max.size, ]
+         no.events[i, 1] <- nrow(tmp[tmp[, 6] == 1, ])
+         no.events[i, 2] <- mean(tmp[tmp[, 6] == 1, 4] - tmp[tmp[,
+             6] == 1, 3])
+         no.events[i, 3] <- nrow(tmp[tmp[, 6] == 2, ])
+         no.events[i, 4] <- mean(tmp[tmp[, 6] == 2, 4] - tmp[tmp[,
+             6] == 2, 3])
+         no.events[i, 5] <- nrow(tmp[tmp[, 6] == 3, ])
+         no.events[i, tmp[tmp[, 6] == 3, 2]] <- 1
+     }
+     return(no.events)
```

```
+ }
```

This function defines and summarizes the regions of telomeric copy number changes

```
> no.tel.cna <- function(seg.out, chrominfo, min.size = 0, min.probes = 1,
+     max.size = 1e+09, gain = log2(2.5/2), loss = log2(1.5/2)) {
+     if (class(seg.out) == "DNAcopy") {
+         seg.out <- seg.out$output
+     }
+     tmp.segs <- seg.out
+     tmp.segs <- tmp.segs[!tmp.segs[, 5] < min.probes, ]
+     tmp.segs[, 2] <- as.character(tmp.segs[, 2])
+     if (nrow(tmp.segs[tmp.segs[, 2] == "X", ]) > 0) {
+         tmp.segs[tmp.segs[, 2] == "X", 2] <- rep(23, nrow(tmp.segs[tmp.segs[,
+             2] == "X", ]))
+     }
+     seg.out <- seg.out[!seg.out[, 2] %in% c("MT", "Y", "24"),
+         ]
+     tmp.segs[tmp.segs[, 6] >= gain, 6] <- 10
+     tmp.segs[tmp.segs[, 6] <= loss, 6] <- -10
+     tmp.segs[tmp.segs[, 6] > loss & tmp.segs[, 6] < gain, 6] <- 0
+     for (j in 1:length(unique(tmp.segs[, 1]))) {
+         tmp.sample <- tmp.segs[tmp.segs[, 1] == unique(tmp.segs[,
+             1])[j], ]
+         for (i in 1:23) {
+             tmp1 <- tmp.sample[tmp.sample[, 2] == i, , drop = F]
+             if (tmp1[1, 6] == 10 & nrow(tmp1) != 1 & tmp1[1,
+                 4] < (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 1
+             }
+             if (tmp1[nrow(tmp1), 6] == 10 & nrow(tmp1) != 1 &
+                 tmp1[nrow(tmp1), 3] > (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][nrow(tmp.sample[tmp.sample[,
+                   2] == i, ])] <- 1
+             }
+             if (tmp1[1, 6] == -10 & nrow(tmp1) != 1 & tmp1[1,
+                 4] < (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 2
+             }
+             if (tmp1[nrow(tmp1), 6] == -10 & nrow(tmp1) != 1 &
+                 tmp1[nrow(tmp1), 3] > (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][nrow(tmp.sample[tmp.sample[,
+                   2] == i, ])] <- 2
+             }
+             if (nrow(tmp.sample[tmp.sample[, 2] == i, ]) == 10 &
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] != 0) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 3
+             }
+             if (nrow(tmp.sample[tmp.sample[, 2] == i, ]) == -10 &
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] != 0) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 4
+             }
+         }
+         tmp.segs[tmp.segs[, 1] == unique(tmp.segs[, 1])[j], ] <- tmp.sample
```

```
+     }
+     no.events <- matrix(0, nrow = length(unique(tmp.segs[, 1])),
+         ncol = 9)
+     rownames(no.events) <- unique(tmp.segs[, 1])
+     colnames(no.events) <- c("Total telomeric CNA", "Mean size",
+         "Telomeric gain", "Mean size", "Telomeric loss", "Mean size",
+         "Total whole chromosome CNA", "Whole chromosome gain",
+         "Whole chromosome loss")
+     for (i in unique(tmp.segs[, 1])) {
+         tmp <- tmp.segs[tmp.segs[, 1] == i, ]
+         tmp <- tmp[(tmp[, 4] - tmp[, 3]) > min.size, ]
+         tmp <- tmp[(tmp[, 4] - tmp[, 3]) < max.size, ]
+         no.events[i, 1] <- nrow(tmp[tmp[, 6] == 1 | tmp[, 6] ==
+             2, ])
+         no.events[i, 2] <- mean(tmp[tmp[, 6] == 1 | tmp[, 6] ==
+             2, 4] - tmp[tmp[, 6] == 1 | tmp[, 6] == 2, 3])
+         no.events[i, 3] <- nrow(tmp[tmp[, 6] == 1, ])
+         no.events[i, 4] <- mean(tmp[tmp[, 6] == 1, 4] - tmp[tmp[,
+             6] == 1, 3])
+         no.events[i, 5] <- nrow(tmp[tmp[, 6] == 2, ])
+         no.events[i, 6] <- mean(tmp[tmp[, 6] == 2, 4] - tmp[tmp[,
+             6] == 2, 3])
+         no.events[i, 7] <- nrow(tmp[tmp[, 6] == 3 | tmp[, 6] ==
+             4, ])
+         no.events[i, 8] <- nrow(tmp[tmp[, 6] == 3, ])
+         no.events[i, 9] <- nrow(tmp[tmp[, 6] == 4, ])
+     }
+     return(no.events)
+ }
```

This function counts the total number of copy number aberrations

```
> gaps <- function(cbs.out, min.size = 0, gain = log2(2.5/2), loss = log2(1.5/2),
+     min.probes = 1) {
+     if (length(gain) == 1) {
+         gain <- rep(gain, length(unique(cbs.out[, 1])))
+     }
+     if (length(loss) == 1) {
+         loss <- rep(loss, length(unique(cbs.out[, 1])))
+     }
+     stopifnot(length(gain) == length(unique(cbs.out[, 1])))
+     stopifnot(length(loss) == length(unique(cbs.out[, 1])))
+     cbs.out <- cbs.out[!(cbs.out[, 2] == 24), ]
+     cbs.out <- cbs.out[!(cbs.out[, 5] <= min.probes), ]
+     gaps <- matrix(rep(NA, length(unique(cbs.out[, 1]))), nrow = length(unique(cbs.out[,
+         1])), ncol = 7)
+     rownames(gaps) <- unique(cbs.out[, 1])
+     colnames(gaps) <- c("Short Aberrations", "Long Aberrations",
+         "Short gains", "Short deletions", "Total Gains", "Total Deletions",
+         "Total Aberrations")
+     for (i in 1:nrow(gaps)) {
+         a <- cbs.out[cbs.out[, 1] == rownames(gaps)[i], ]
+         a <- a[a[, 6] >= gain[i] | a[, 6] <= loss[i], ]
+         gaps[i, 1] <- nrow(a[a[, 4] - a[, 3] <= min.size, ])
```

```
+         gaps[i, 2] <- nrow(a[a[, 4] - a[, 3] > min.size, ])
+         gaps[i, 5] <- nrow(a[a[, 6] > gain[i], ])
+         gaps[i, 6] <- nrow(a[a[, 6] < loss[i], ])
+         b <- a[a[, 4] - a[, 3] <= min.size, ]
+         gaps[i, 3] <- nrow(b[b[, 6] > gain[i], ])
+         gaps[i, 4] <- nrow(b[b[, 6] < loss[i], ])
+     }
+     gaps[, 7] <- gaps[, 5] + gaps[, 6]
+     return(gaps)
+ }
```

This function identifies all telomeric AI and returns the location

```
> telAI.fn <- function(seg.out, chrominfo, output = "tel", min.size = 0,
+     min.probes = 1, max.size = 1e+09) {
+     if (class(seg.out) == "DNAcopy") {
+         seg.out <- seg.out$output
+     }
+     tmp.segs <- seg.out[!seg.out[, 2] %in% c("MT", "Y", "24"),
+         ]
+     tmp.segs <- tmp.segs[!tmp.segs[, 5] < min.probes, ]
+     tmp.segs[, 2] <- as.character(tmp.segs[, 2])
+     if (nrow(tmp.segs[tmp.segs[, 2] == "X", ]) > 0) {
+         tmp.segs[tmp.segs[, 2] == "X", 2] <- rep(23, nrow(tmp.segs[tmp.segs[,
+             2] == "X", ]))
+     }
+     tmp.segs[tmp.segs[, 6] == 1, 6] <- 2
+     tmp.segs <- tmp.segs[tmp.segs[, 4] - tmp.segs[, 3] >= min.size,
+         ]
+     for (j in 1:length(unique(tmp.segs[, 1]))) {
+         tmp.sample <- tmp.segs[tmp.segs[, 1] == unique(tmp.segs[,
+             1])[j], ]
+         for (i in 1:23) {
+             tmp1 <- tmp.sample[tmp.sample[, 2] == i, , drop = F]
+             if (tmp1[1, 6] == 2 & nrow(tmp1) != 1 & tmp1[1, 4] <
+                 (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 1
+             }
+             if (tmp1[nrow(tmp1), 6] == 2 & nrow(tmp1) != 1 &
+                 tmp1[nrow(tmp1), 3] > (chrominfo[i, 3] * 1000)) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][nrow(tmp.sample[tmp.sample[,
+                   2] == i, ])] <- 1
+             }
+             if (nrow(tmp.sample[tmp.sample[, 2] == i, ]) == 1 &
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] != 0) {
+                 tmp.sample[tmp.sample[, 2] == i, 6][1] <- 3
+             }
+         }
+         tmp.segs[tmp.segs[, 1] == unique(tmp.segs[, 1])[j], ] <- tmp.sample
+     }
+     if (output == "tel") {
+         tmp.segs <- tmp.segs[tmp.segs[, 6] == 1, ]
+     }
+     if (output == "int") {
```

```
+         tmp.segs <- tmp.segs[tmp.segs[, 6] == 2, ]
+     }
+     return(tmp.segs)
+ }
```

This function takes as input breakpoint location of telomeric AI, and then test each breakpoint for association with a given DNA structure, supplied as a two-column matrix of locations

```
> telBreak.CNV.fn <- function(test.breaks, test.loc, window.size = 25000) {
+     break.loc <- matrix(0, 0, 2)
+     for (i in 1:nrow(test.breaks)) {
+         break.loc <- rbind(break.loc, c(as.numeric(test.breaks[i,
+             2]), NA))
+         if (as.numeric(test.breaks[i, 3]) < chrominfo[as.numeric(test.breaks[i,
+             2]), 3] * 1000) {
+             break.loc[i, 2] <- as.numeric(test.breaks[i, 4])
+         }
+         if (as.numeric(test.breaks[i, 3]) > chrominfo[as.numeric(test.breaks[i,
+             2]), 3] * 1000) {
+             break.loc[i, 2] <- as.numeric(test.breaks[i, 3])
+         }
+     }
+     test.breaksDNA <- vector()
+     for (i in 1:nrow(break.loc)) {
+         tmp <- test.loc[test.loc[, 1] == break.loc[i, 1], ]
+         a <- c(break.loc[i, 2] - window.size, break.loc[i, 2] +
+             window.size)
+         tmp <- tmp[tmp[, 3] >= a[1], , drop = F]
+         tmp <- tmp[tmp[, 2] <= a[2], , drop = F]
+         test.breaksDNA[i] <- nrow(tmp)
+     }
+     tmp <- c(length(test.breaksDNA[test.breaksDNA > 0]), length(test.breaksDNA))
+     names(tmp) <- c("Associated with", "Total")
+     return(tmp)
+ }
```

This functions calculate sensitivity, specificity, positive predictive value, negative predictive value, accuracy, and a p-value, based on predicted versus true input

```
> acc.calc <- function(test, truth) {
+     if (class(test) == "logical") {
+         test <- c(0, 1)[match(test, c("FALSE", "TRUE"))]
+     }
+     if (class(truth) == "logical") {
+         truth <- c(0, 1)[match(truth, c("FALSE", "TRUE"))]
+     }
+     TN <- length(test[test == 0 & truth == 0])
+     TP <- length(test[test == 1 & truth == 1])
+     FN <- length(test[test == 0 & truth == 1])
+     FP <- length(test[test == 1 & truth == 0])
+     c(ACC = (TP + TN)/(TP + FP + FN + TN), PPV = TP/(TP + FP),
+         NPV = TN/(TN + FN), SENS = TP/(TP + FN), SPEC = TN/(TN +
+             FP), P = fisher.test(table(test, truth))$p.value)
+ }
```

This is a convenience function that simply takes two vectors as input, and add one as names for the other

```
> name.vect <- function(x, y) {
+     names(x) <- y
+     return(x)
+ }
```

These two functions are used to calculate confidence intervals for AUC curves by bootstrapping 1000 times

```
> AUC.stat <- function(data, indices) {
+     d <- data[indices, ]
+     return(AUC(rocdemo.sca(d[, 1], d[, 2])))
+ }
> ci.auc <- function(data, pcr, subset = TRUE) {
+     data <- data[subset]
+     pcr <- pcr[subset]
+     results <- vector()
+     d <- cbind(pcr, data)
+     tmp <- boot(data = d, statistic = AUC.stat, R = 1000)
+     tmp <- boot.ci(tmp, type = "bca")
+     results[1] <- tmp$bca[4]
+     results[2] <- tmp$bca[5]
+     names(results) <- c("Low", "High")
+     return(results)
+ }
```

2 Load and organize data

Load a few necessary objects

The first is an object that holds information about chromosome size and location of the centromere It can be reconstructed by loading a 23x3 matrix with chromosome number in the first column, chromosome length in the second, and centromere location in the third.

```
> load("chrominfo.RData")
> head(chrominfo)
```

```
  chrom   length centromere
1     1 245522.8  122267.48
2     2 243018.2   93248.04
3     3 199505.7   92037.54
4     4 191411.2   51001.04
5     5 180857.9   47941.40
6     6 170975.7   60438.12
```

2.1 Load cell line data

This is an object with segmented copy numbers of the cell lines, based on SNP6 data obtained from the Sanger center.

Copy number was determined based on data pre-processed using the aroma.affymetrix CRMAv2 and CalMaTe algorithms, segmentation was performed using the R package DNAcopy. The object can be reconstructed by downloading the cell line data from the Welcome Trust fund, preprocessing via the aroma packages, and segmenting the output using the R package DNAcopy, with default parameters.

```
> load("cellLines.CN.seg.aroma.20111220-1.RData")
> head(cellLines.CN.seg)
```

```
     ID chrom loc.start  loc.end num.mark seg.mean
1 BT-20     1     51599    88466       18  -0.7428
2 BT-20     1    218570   346406       12   0.3422
3 BT-20     1    394035 16799099     9024  -0.0542
4 BT-20     1  16803917 16878322       38   0.2721
5 BT-20     1  16878364 16888847       17  -0.3034
6 BT-20     1  16889460 16984147       42   0.2651
```

This is the output from the ASCAT v2 algorithm (http://heim.ifi.uio.no/bioinf/Projects/ASCAT/), based on the log2 ratios and B-allele frequencies. To reconstruct this object, retrieve the cell line Affymetrix SNP6 data from Sanger, and calculate and extract the log2 ratios and BAF. We used the aroma.affymetrix and calmate packages for pre-processing, available at www.aroma-project.org.

```
> load("cellLines.ascat.output.RData")
> str(cellLines.ascat.output)
```

```
List of 7
 $ nA                  : num [1:1868855, 1:18] 2 2 2 2 2 2 2 2 2 2 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1868855] "CN_473963" "CN_473964" "CN_473965" "CN_477984" ...
  .. ..$ : chr [1:18] "BT-20_4218" "BT-549_3943" "CAMA-1_3659" "HCC1143-BL_4099" ...
 $ nB                  : num [1:1868855, 1:18] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
```

```
 .. ..$ : chr [1:1868855] "CN_473963" "CN_473964" "CN_473965" "CN_477984" ...
 .. ..$ : chr [1:18] "BT-20_4218" "BT-549_3943" "CAMA-1_3659" "HCC1143-BL_4099" ...
$ aberrantcellfraction: num [1:18] 0.75 0.73 0.64 0.22 0.7 1 0.74 0.6 0.87 0.57 ...
$ ploidy              : num [1:18] 2.98 4.1 3.75 4.61 4.4 ...
$ psi                 : num [1:18] 2.9 4.2 3.95 4.25 3.9 1.95 2.35 2.15 2.1 2.3 ...
$ failedarrays        : chr [1:3] "BT-474_3712" "HCC38_3392" "MDA-MB-453_4502"
$ segments            :List of 21
 ..$ : num [1:332, 1:4] 1 28883 30031 34322 36936 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : logi NA
 ..$ : num [1:131, 1:4] 1 8871 12881 75901 118519 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:197, 1:4] 1 17622 24427 51160 51503 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:64, 1:4] 1 75901 107754 108890 146402 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:311, 1:4] 1 3031 11311 13411 14708 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:34, 1:4] 1 146402 300065 427831 449839 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:206, 1:4] 1 5810 17701 17763 51728 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:50, 1:4] 1 75901 78879 146402 184612 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:44, 1:4] 1 44212 44260 146402 207847 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:164, 1:4] 1 33510 36391 51728 75901 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:64, 1:4] 1 75901 79304 106722 107097 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : NULL
 .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
 ..$ : num [1:333, 1:4] 1 74020 74442 75901 77875 ...
 .. ..- attr(*, "dimnames")=List of 2
```

```
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:51, 1:4] 1 14094 15710 78907 78949 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : logi NA
  ..$ : num [1:391, 1:4] 1 66244 67087 67256 67849 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:142, 1:4] 1 2658 3950 15934 18540 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:374, 1:4] 1 5756 6012 28835 30179 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : logi NA
  ..$ : num [1:172, 1:4] 1 712 913 6649 7117 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:169, 1:4] 1 23222 23257 75901 79124 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
```

Load the probe locations

```
> load("snp6.markers.ascat.RData")
> head(snp6.markers.ascat)

          Chrom Position
CN_473963     1    51599
CN_473964     1    51672
CN_473965     1    51687
CN_477984     1    52016
CN_473981     1    52784
CN_473982     1    52801
```

This is the cisplatin IC50 values

```
> load("ic50.CL.20111107.RData")
> ic50.CL

           Cisplatin IC50 Subtype
BT-20                9.85    TNBC
BT-474              35.89    HER2
BT-549              21.01    TNBC
CAMA-1              40.82      ER
HCC1143             16.71    TNBC
HCC1187             21.25    TNBC
HCC1954             37.93    HER2
```

```
HCC38               12.02    TNBC
MCF7                54.71      ER
MDA-MB-231          37.05    TNBC
MDA-MB-361          32.31    HER2
MDA-MB-453          50.14    TNBC
MDA-MB-468          10.40    TNBC
T47D                 3.85      ER
```

Now organize the output from ASCAT, to get it into a more compact format

```
> cellLines.ascat.seg <- organize.ascat.segments(cellLines.ascat.output,
+     snp6.markers.ascat)
> head(cellLines.ascat.seg)
```

```
    SampleID Chr    Start      End nProbes nA nB
1 BT-20_4218   1    51599 50616134   28882  1  1
2 BT-20_4218   1 50616469 53199029    1148  2  0
3 BT-20_4218   1 53199571 58595853    4291  1  1
4 BT-20_4218   1 58597518 62161587    2614  2  1
5 BT-20_4218   1 62161809 63631353    1052  2  0
6 BT-20_4218   1 63631634 66046745    1904  1  1
```

Remove the numeric ID part of the cell line names

```
> cellLines.ascat.seg[, 1] <- sub("(_\\d+$)", "", cellLines.ascat.seg[,
+     1])
```

Now we call allelic imbalance based on the output from ASCAT. Any time the two alleles are not represented in equal amounts (balanced), that is defined as AI

```
> AI <- c(0, 1)[match(cellLines.ascat.seg[, 6] == cellLines.ascat.seg[,
+     7], c("TRUE", "FALSE"))]
> cellLines.ascat.seg <- cbind(cellLines.ascat.seg[, 1:5], AI,
+     cellLines.ascat.seg[, 6:7])
> head(cellLines.ascat.seg)
```

```
  SampleID Chr    Start      End nProbes AI nA nB
1    BT-20   1    51599 50616134   28882  0  1  1
2    BT-20   1 50616469 53199029    1148  1  2  0
3    BT-20   1 53199571 58595853    4291  0  1  1
4    BT-20   1 58597518 62161587    2614  1  2  1
5    BT-20   1 62161809 63631353    1052  1  2  0
6    BT-20   1 63631634 66046745    1904  0  1  1
```

Check if any samples failed ASCAT

```
> cellLines.ascat.output$failedarrays
```

```
[1] "BT-474_3712"     "HCC38_3392"      "MDA-MB-453_4502"
```

three failed

Analyze the cell lines data for AI

```
> min.probes.cl <- 500
> indx.cl <- intersect(rownames(ic50.CL), cellLines.ascat.seg[,
+     1])
```

As MCF7 is a clear outlier in our data, very drug resistant, and it is reported as caspase 3 deficient, this sample is removed.

```
> indx.cl <- indx.cl[!indx.cl == "MCF7"]
> ic50.CL <- ic50.CL[indx.cl, ]
> no.tel.CL <- no.tel(cellLines.ascat.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cl)[indx.cl, ]
> head(no.tel.CL)
```

```
        Telomeric AI Mean size Interstitial AI Mean Size Wholo chr AI 1 2 3 4 5
BT-20             33  19885273             188   9720339            1 0 0 0 0 0
BT-549            27  36632057              62  22282507            3 0 0 0 0 0
CAMA-1            22  28174680              98   9820586            0 0 0 0 0 0
HCC1143           32  18706803             189   9971368            0 0 0 0 0 0
HCC1187           30  21998968              82  18436058            2 0 0 0 0 0
HCC1954           25  27968028             156   7820720            0 0 0 0 0 0
        6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
BT-20   0 0 0 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
BT-549  0 0 0 0  0  0  0  0  0  0  0  0  0  1  0  1  1  0
CAMA-1  0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
HCC1143 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
HCC1187 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  1  1  0
HCC1954 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
> no.tel.cna.CL <- no.tel.cna(cellLines.CN.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cl)[indx.cl, ]
> head(no.tel.cna.CL)
```

```
        Total telomeric CNA Mean size Telomeric gain Mean size Telomeric loss
BT-20                    12   7359561              6   6959751              6
BT-549                    4   8708811              2   2496946              2
CAMA-1                    2   4307014              2   4307014              0
HCC1143                  12  11261142              9   6731372              3
HCC1187                   6   2958636              6   2958636              0
HCC1954                   7   5531236              6   4435542              1
        Mean size Total whole chromosome CNA Whole chromosome gain
BT-20     7759371                         0                     0
BT-549   14920676                         0                     0
CAMA-1        NaN                         0                     0
HCC1143  24850450                         1                     1
HCC1187       NaN                         0                     0
HCC1954  12105397                         0                     0
        Whole chromosome loss
BT-20                       0
BT-549                      0
CAMA-1                      0
HCC1143                     0
HCC1187                     0
HCC1954                     0
```

```
> no.cna.CL <- gaps(cellLines.CN.seg, min.probes = min.probes.cl)[indx.cl,
+     ]
> head(no.cna.CL)
```

```
        Short Aberrations Long Aberrations Short gains Short deletions
BT-20                   0              87           0               0
```

BT-549	0	41	0	0
CAMA-1	0	35	0	0
HCC1143	0	106	0	0
HCC1187	0	77	0	0
HCC1954	0	80	0	0

	Total Gains	Total Deletions	Total Aberrations
BT-20	61	26	87
BT-549	24	17	41
CAMA-1	22	13	35
HCC1143	93	13	106
HCC1187	66	11	77
HCC1954	68	12	80

2.2 Load the Heiser et al cell line data

This is the cell line data from the Heiser et al, 2011, PNAS paper. The SNP data was pre-processed in the same manner as the SNP data from the Sanger center

Here we load the output from ASCAT

```
> load("heiser_cl.ascat_out.RData")
> str(heiser_cl.ascat_out)

List of 7
 $ nA                  : num [1:1868855, 1:37] 2 2 2 2 2 2 2 2 2 2 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1868855] "CN_473963" "CN_473964" "CN_473965" "CN_477984" ...
  .. ..$ : chr [1:37] "080122_SNP6.0_184B5_B01" "080122_SNP6.0_185A1_A01" "080122_SNP6.0_600MPE_C
 $ nB                  : num [1:1868855, 1:37] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1868855] "CN_473963" "CN_473964" "CN_473965" "CN_477984" ...
  .. ..$ : chr [1:37] "080122_SNP6.0_184B5_B01" "080122_SNP6.0_185A1_A01" "080122_SNP6.0_600MPE_C
 $ aberrantcellfraction: num [1:37] 0.9 0.9 0.85 0.83 0.78 1 1 1 1 0.74 ...
 $ ploidy              : num [1:37] 2.26 1.97 2.08 4.14 2.56 ...
 $ psi                 : num [1:37] 2.4 1.95 1.95 3.85 2.45 2.85 2.95 3.5 3.9 3.25 ...
 $ failedarrays        : chr [1:16] "080122_SNP6.0_BT549_G01" "080122_SNP6.0_HCC1143_E02" "080122
 $ segments            :List of 53
  ..$ : num [1:114, 1:4] 1 77462 109120 146402 171153 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:69, 1:4] 1 78477 79359 110296 111287 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:145, 1:4] 1 8899 14959 15110 16138 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:419, 1:4] 1 32336 75901 79781 80383 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:336, 1:4] 1 19835 20320 28914 30091 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:154, 1:4] 1 83214 88610 89775 95631 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:117, 1:4] 1 15955 41860 68828 70056 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : logi NA
  ..$ : num [1:114, 1:4] 1 6198 6253 24423 51165 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
```

```
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:128, 1:4] 1 5807 17708 17761 51986 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:318, 1:4] 1 4938 4961 20801 22308 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:221, 1:4] 1 29047 29918 31992 32164 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:252, 1:4] 1 525 5176 6482 7420 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:152, 1:4] 1 15827 30399 146402 152553 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : logi NA
..$ : num [1:300, 1:4] 1 75901 77876 81160 81825 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:414, 1:4] 1 16135 36693 37254 53094 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:167, 1:4] 1 17972 18041 75901 76240 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:305, 1:4] 1 3834 15414 32347 32791 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:455, 1:4] 1 7909 9232 18262 19216 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:179, 1:4] 1 2556 14470 75901 100034 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:371, 1:4] 1 38994 40403 66245 67084 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
```

```
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:32, 1:4] 1 111859 146402 300065 301116 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:191, 1:4] 1 16116 22288 38574 75901 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:256, 1:4] 1 53125 60019 61832 66244 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:108, 1:4] 1 44490 75901 146402 161960 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:210, 1:4] 1 44212 44260 75901 78909 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:497, 1:4] 1 4144 4298 11971 12412 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : logi NA
..$ : num [1:97, 1:4] 1 536 20013 20699 73009 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:225, 1:4] 1 5152 22808 31153 32030 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:144, 1:4] 1 60048 60862 79124 79160 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:128, 1:4] 1 15213 66409 67209 74976 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : logi NA
..$ : logi NA
..$ : num [1:136, 1:4] 1 12191 58055 60974 61047 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
```

```
  ..$ : num [1:327, 1:4] 1 47071 51679 75901 96983 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:149, 1:4] 1 15791 23229 23256 75901 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:222, 1:4] 1 15343 15524 28708 28782 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:253, 1:4] 1 2581 3462 4029 4239 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:267, 1:4] 1 2532 3450 4029 4234 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:299, 1:4] 1 8467 13341 13995 43560 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
```

Now we load the cell line subtypes, as defined in the original paper. This is supplementary table 1

```
> heiser.cl.sub.mat <- read.xls("heiser.sd01.xls", skip = 1, header = T,
+     row.names = 1, as.is = T)
> heiser.cl.sub <- heiser.cl.sub.mat[1:54, 2]
> names(heiser.cl.sub) <- rownames(heiser.cl.sub.mat[1:54, ])
> heiser.cl.sub
```

```
                184A1                 184B5                600MPE
"Non-malignant, Basal" "Non-malignant, Basal"             "Luminal"
                AU565                  BT20                 BT474
            "ERBB2AMP"               "Basal"            "ERBB2AMP"
                BT483                 BT549                 CAMA1
             "Luminal"         "Claudin-low"             "Luminal"
              HCC1143               HCC1187              HCC1395
              "Basal"               "Basal"         "Claudin-low"
              HCC1419               HCC1428               HCC1599
            "ERBB2AMP"             "Luminal"               "Basal"
              HCC1806               HCC1937               HCC1954
              "Basal"               "Basal"            "ERBB2AMP"
               HCC202               HCC2185              HCC2218
            "ERBB2AMP"             "Luminal"            "ERBB2AMP"
              HCC3153                 HCC38                HCC70
              "Basal"         "Claudin-low"               "Basal"
               HS578T                   LY2                MCF10A
       "Claudin-low"             "Luminal" "Non-malignant, Basal"
               MCF10F                MCF12A                 MCF7
"Non-malignant, Basal" "Non-malignant, Basal"             "Luminal"
           MDAMB134VI               MDAMB157           MDAMB175VII
```

```
          "Luminal"          "Claudin-low"              "Luminal"
           MDAMB231               MDAMB361               MDAMB415
      "Claudin-low"             "ERBB2AMP"              "Luminal"
           MDAMB436               MDAMB453               MDAMB468
      "Claudin-low"              "Luminal"                "Basal"
              SKBR3               SUM102PT             SUM1315MO2
         "ERBB2AMP"                "Basal"          "Claudin-low"
           SUM149PT               SUM159PT               SUM185PE
            "Basal"          "Claudin-low"              "Luminal"
           SUM225CWN                SUM44PE                SUM52PE
         "ERBB2AMP"              "Luminal"              "Luminal"
                T47D               UACC812                UACC893
           "Luminal"             "ERBB2AMP"              "Luminal"
               ZR751                  ZR7530                  ZR75B
           "Luminal"              "Luminal"              "Luminal"
```

This is the drug GI50 values from the paper.

```
> heiser.cl.gi50 <- read.xls("heiser.sd02.xlsx", skip = 1, header = T,
+     row.names = 1, as.is = T)
> tmp <- rownames(heiser.cl.gi50)
> heiser.cl.gi50 <- apply(heiser.cl.gi50, 2, as.numeric)
> rownames(heiser.cl.gi50) <- tmp
> head(heiser.cl.gi50)
```

```
        X17.AAG  X5.FdUR    X5.FU   AG1024   AG1478 Sigma.AKT1.2.inhibitor
600MPE 6.870066 4.114703       NA       NA 3.992638                     NA
AU565  7.247734 5.183882 4.972114 4.477121 4.568730               5.609844
BT20         NA       NA 3.487703 4.477121       NA               5.004351
BT474  7.690189 3.172659 3.290527 4.477121 6.174157               6.080043
BT483  6.654451 4.480107 4.134974 4.477121 5.638744               6.082212
BT549  7.472463 3.735296       NA 4.477121 4.410818                     NA
       Triciribine AS.252424 AZD6244   BEZ235 BIBW2992 Bortezomib Carboplatin
600MPE    5.434484        NA      NA       NA       NA   6.372983    3.817416
AU565     6.801077  4.874750      NA 6.591219       NA   8.275530    4.943955
BT20      5.260587  4.648175 4.30103 5.420143 5.558814   7.327165          NA
BT474     6.399033  5.362300 4.30103 6.459627 8.232475   8.130542    3.976576
BT483     6.906122  5.365980 4.30103 4.949954 5.779160   7.708806    5.822004
BT549     4.226665        NA      NA       NA       NA   8.215942    4.578513
       CGC.11047 CGC.11144 Cisplatin   CPT.11 Docetaxel Doxorubicin Epirubicin
600MPE  3.327431  6.485800  4.331474 4.677356  7.009101    6.568268   6.462843
AU565   3.543755  6.311005  5.728188 5.907733  8.277936    7.025223   6.843361
BT20          NA  6.520302        NA       NA        NA          NA         NA
BT474   3.569146  6.022828  4.475537 4.108876  8.198231    6.511953   5.170984
BT483   3.228546  6.251072  3.588514 5.332785  7.633831    6.815615   6.782119
BT549   4.533542  6.652128  5.418136       NA        NA          NA   6.694356
       Erlotinib Etoposide Fascaplysin Geldanamycin Gemcitabine Glycyl.H.1152
600MPE  4.281280  5.011165    6.544616     7.414693    7.638963            NA
AU565   4.884756  6.174357    6.920960     7.287818    7.808269      5.141460
BT20    5.696155  5.475872    6.506515           NA          NA      5.146230
BT474   4.977173  4.723689    6.720573     7.841636    3.978604      4.176091
BT483   4.176091  5.370604    7.182715     6.839121    8.050328      4.353635
BT549   4.378272  5.856931    6.286934     8.260421    8.173408            NA
       GSK923295 Lapatinib GSK1070916 GSK1120212  TGX.221 GSK1838705 GSK461364
600MPE  4.477121  4.778151   5.102796   8.166786 5.088270   6.485846  5.161191
```

```
AU565   7.615600  6.397368   5.517298   4.821626 5.181884   5.627646  8.348481
BT20          NA  4.778151         NA         NA 4.774239   4.634534        NA
BT474   5.418659  6.402238   5.194975   4.778151 5.099236   5.079197  5.069412
BT483   6.444275  4.778151   5.349720   4.778151 5.367208   5.515607  5.354183
BT549         NA  4.778151         NA   5.166945 4.615909   5.210978        NA
       GSK2119563 GSK2126458 GSK1487371 GSK1059615 Ibandronate.sodium.salt
600MPE   6.229963   8.220457         NA   6.310266                      NA
AU565    6.253141   8.099501   5.892357   6.316662                3.744760
BT20     5.968889   7.796358   4.176091         NA                4.685680
BT474    6.821881   8.361199         NA   6.800504                3.980996
BT483    7.465795   8.938289   5.569299         NA                4.239874
BT549    5.383742   7.317398   5.453782   5.730035                      NA
       ICRF.193 Gefitinib Ixabepilone   LBH589 Lestaurtinib Methotrexate
600MPE       NA  5.137746    5.282331 6.728689     5.770670     3.778086
AU565  6.138037  5.973781    8.369717 6.978742     6.072466     3.778076
BT20   4.380211        NA    8.085915 6.413439     5.486464     3.477121
BT474  4.301030  6.142289    8.078522 7.456178     6.608762     3.477121
BT483        NA  5.207217    5.267766 7.136833     6.126210           NA
BT549        NA  4.815894    8.221909       NA           NA     3.477121
        MLN4924 NSC.663284 Nutlin.3a   NU6102 Oxaliplatin Oxamflatin Paclitaxel
600MPE 6.425884   5.335674  4.323718       NA    4.889908         NA   7.184148
AU565  6.738754   5.807986  4.789681 4.639439    5.545216   6.189237   8.085250
BT20   5.563240   5.476108  4.466768 4.231694          NA   5.416749         NA
BT474  6.244667   5.563850  4.389384 4.555317    4.731504   6.567702   7.991164
BT483  4.476286   6.016736  5.191499 4.176091    4.560631   6.146007   7.460540
BT549        NA         NA  4.345300       NA    5.716196         NA         NA
       PD173074 PD.98059 Pemetrexed Purvalanol.A L.779450 Rapamycin Vorinostat
600MPE 5.010740 4.301030         NA     4.519392       NA        NA   4.153605
AU565  5.125665 5.124362   2.533948     5.009951 4.477121  7.498848   4.075512
BT20   4.795618 4.044107         NA     4.564674 4.436846  7.869113   3.716947
BT474  4.477121 4.000000   2.533948     3.778151 4.727404  7.816706   4.262598
BT483        NA 4.124939   2.533948     4.402312 4.839597  8.777219   4.227087
BT549  5.133112       NA   2.533948     3.778151       NA  4.477121   3.828892
         SB.3CT Ispinesib Bosutinib Sorafenib Sunitinib.Malate Tamoxifen
600MPE       NA  7.679567  5.053721  4.342794         5.366754  4.320089
AU565  4.000000  7.649836  5.674507  3.750202         5.423731  4.537336
BT20   4.423512  7.771662  5.862346  4.202799         4.784269        NA
BT474  4.985933  7.290497  6.141099  3.997965         4.765132  5.621945
BT483  4.590438 10.313162  5.449399  4.925334         4.729170  4.621294
BT549        NA  7.332658        NA  3.921788         5.288711  3.778151
       TCS.JNK.5a TCS.2312.dihydrochloride Temsirolimus   TPCA.1 Topotecan
600MPE         NA                6.216676     4.743841 4.176091        NA
AU565          NA                6.563285     6.996419 4.176091  7.725173
BT20     5.970907                5.703272     6.113962 4.356667        NA
BT474    4.174785                6.210980     7.869440 4.176091  5.597722
BT483    5.937292                6.178364     4.176090       NA  7.792461
BT549          NA                      NA           NA 4.176091        NA
       Trichostatin.A Vinorelbine   VX.680   XRP44X ZM.447439
600MPE       5.184590    5.289934       NA       NA        NA
AU565        5.429726    8.062586 5.663896 6.353525  5.820232
BT20         4.806435          NA 4.722649 5.291695  5.286465
BT474        5.002392    7.322115 4.544742 5.342283  4.200410
BT483        5.000124    8.141970 5.441521 4.175240  4.568856
BT549        5.131921    8.021364       NA       NA        NA
```

Now organize the output from ASCAT, to get it into a more compact format

```
> heiser.cl.ascat.seg <- organize.ascat.segments(heiser_cl.ascat_out,
+     snp6.markers.ascat)
> head(heiser.cl.ascat.seg)
```

```
                SampleID Chr     Start       End nProbes nA nB
1 080122_SNP6.0_184B5_B01   1     51599 148116976   77461  1  1
2 080122_SNP6.0_184B5_B01   1 148117445 194506255   31658  2  1
3 080122_SNP6.0_184B5_B01   1 194506743 247191012   37282  3  1
4 080122_SNP6.0_184B5_B01   2      2785  34544573   24751  1  1
5 080122_SNP6.0_184B5_B01   2  34552819  34590562      47  0  0
6 080122_SNP6.0_184B5_B01   2  34590667 137128697   62331  1  1
```

Now we call allelic imbalance based on the output from ASCAT. Any time the two alleles are not represented in equal amounts (balanced), that is defined as AI

```
> AI <- c(0, 1)[match(heiser.cl.ascat.seg[, 6] == heiser.cl.ascat.seg[,
+     7], c("TRUE", "FALSE"))]
> heiser.cl.ascat.seg <- cbind(heiser.cl.ascat.seg[, 1:5], AI,
+     heiser.cl.ascat.seg[, 6:7])
> head(heiser.cl.ascat.seg)
```

```
                SampleID Chr     Start       End nProbes AI nA nB
1 080122_SNP6.0_184B5_B01   1     51599 148116976   77461  0  1  1
2 080122_SNP6.0_184B5_B01   1 148117445 194506255   31658  1  2  1
3 080122_SNP6.0_184B5_B01   1 194506743 247191012   37282  1  3  1
4 080122_SNP6.0_184B5_B01   2      2785  34544573   24751  0  1  1
5 080122_SNP6.0_184B5_B01   2  34552819  34590562      47  0  0  0
6 080122_SNP6.0_184B5_B01   2  34590667 137128697   62331  0  1  1
```

Check if any samples failed ASCAT

```
> heiser_cl.ascat_out$failedarrays
```

```
 [1] "080122_SNP6.0_BT549_G01"    "080122_SNP6.0_HCC1143_E02"
 [3] "080122_SNP6.0_HCC1500_G02"  "080122_SNP6.0_HCC1599_F06"
 [5] "080122_SNP6.0_HCC1937_G06"  "080122_SNP6.0_HCC2185_H06"
 [7] "080122_SNP6.0_HS578T_C07"   "080122_SNP6.0_MCF10F_E03"
 [9] "080122_SNP6.0_MDAMB157_G03" "080122_SNP6.0_MDAMB361_A04"
[11] "080122_SNP6.0_MDAMB453_E04" "080122_SNP6.0_MDAMB468_F04"
[13] "080122_SNP6.0_SUM1315_F07"  "080122_SNP6.0_SUM185PE_E05"
[15] "080122_SNP6.0_SUM225_F05"   "080122_SNP6.0_SUM229_H07"
```

Sixteen failed, fifteen of these had cisplatin GI50 data

Analyze the cell lines data for AI. As this is also based on SNP6, we use the same minimum number of probes as for the Sanger cell lines

```
> no.tel.heiser.cl <- no.tel(heiser.cl.ascat.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cl)
> head(no.tel.heiser.cl)
```

```
                        Telomeric AI Mean size Interstitial AI Mean Size
080122_SNP6.0_184B5_B01            7  22200782              40   6993984
080122_SNP6.0_185A1_A01            1  60692550              16  17015302
080122_SNP6.0_600MPE_C01           8  26654272              51   4548630
080122_SNP6.0_AU565_B06           29  19050535             204   7137251
```

```
080122_SNP6.0_BT20_D01             32  21868095             165  10684904
080122_SNP6.0_BT474_E01            14  37289921              48  12459545
                       Wholo chr AI 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
080122_SNP6.0_184B5_B01             1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0
080122_SNP6.0_185A1_A01             0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0
080122_SNP6.0_600MPE_C01            0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0
080122_SNP6.0_AU565_B06             0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0
080122_SNP6.0_BT20_D01              0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0
080122_SNP6.0_BT474_E01             1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0
                       18 19 20 21 22 23
080122_SNP6.0_184B5_B01   0  0  1  0  0  0
080122_SNP6.0_185A1_A01   0  0  0  0  0  0
080122_SNP6.0_600MPE_C01  0  0  0  0  0  0
080122_SNP6.0_AU565_B06   0  0  0  0  0  0
080122_SNP6.0_BT20_D01    0  0  0  0  0  0
080122_SNP6.0_BT474_E01   0  0  0  0  1  0
```

Now we fix the names of the SNP files to match the GI50 names

```
> rownames(no.tel.heiser.cl) <- sub("^080122_SNP6.0_", "", rownames(no.tel.heiser.cl))
> rownames(no.tel.heiser.cl) <- sub("^_", "", rownames(no.tel.heiser.cl))
> rownames(no.tel.heiser.cl) <- sub("_\\w+$", "", rownames(no.tel.heiser.cl))
> rownames(no.tel.heiser.cl)

 [1] "184B5"    "185A1"    "600MPE"   "AU565"    "BT20"     "BT474"
 [7] "BT483"    "CAMA1"    "HCC1187"  "HCC1395"  "HCC1419"  "HCC1428"
[13] "HCC1569"  "HCC1954"  "HCC202"   "HCC2218"  "HCC3153"  "HCC38"
[19] "HCC70"    "LY2"      "MCF10A"   "MCF12A"   "MCF7"     "MDAMB231"
[25] "MDAMB415" "MDAMB436" "S1"       "SKBR3"    "SUM102"   "SUM159PT"
[31] "SUM44PE"  "SUM52PE"  "T47D"     "UACC812"  "ZR751"    "ZR75B"
[37] "HCC1806"
```

match the GI50 to the SNP data

```
> indx.heiser <- intersect(rownames(no.tel.heiser.cl), rownames(heiser.cl.gi50))
> indx.heiser

 [1] "600MPE"   "AU565"    "BT20"     "BT474"    "BT483"    "CAMA1"
 [7] "HCC1187"  "HCC1395"  "HCC1419"  "HCC1428"  "HCC1954"  "HCC202"
[13] "HCC3153"  "HCC38"    "HCC70"    "LY2"      "MCF7"     "MDAMB231"
[19] "MDAMB415" "MDAMB436" "SKBR3"    "SUM159PT" "SUM44PE"  "SUM52PE"
[25] "T47D"     "UACC812"  "ZR751"    "ZR75B"    "HCC1806"
```

2.3 Load the Cisplatin-1 trial data

This is the output from ASCAT based on the log2 ratios and B-allele frequencies. To reconstruct this object, retrieve the cisplatin data from GEO, and calculate log2 ratios and BAF either using the submitted processed copy numbers and intensities, or by using the raw cel files.

```
> load("dfci1.ascat.output.RData")
> str(dfci1.ascat.output)

List of 7
 $ nA                  : num [1:50000, 1:27] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:50000] "19503" "50000" "35766" "53523" ...
  .. ..$ : chr [1:27] "X1T" "X3T" "X4T" "X5T" ...
 $ nB                  : num [1:50000, 1:27] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:50000] "19503" "50000" "35766" "53523" ...
  .. ..$ : chr [1:27] "X1" "X3" "X4" "X5" ...
 $ aberrantcellfraction: num [1:27] 0.8 0.71 0.49 0.58 0.69 0.87 0.73 0.71 0.56 0.58 ...
 $ ploidy              : num [1:27] 3.89 4.29 3.8 2.63 1.93 ...
 $ psi                 : num [1:27] 2.4 2.5 3.45 2.05 2.2 1.55 1.65 2.35 2 1.9 ...
 $ failedarrays        : chr(0)
 $ segments            :List of 27
  ..$ : num [1:122, 1:4] 1 134 385 436 666 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:143, 1:4] 1 642 685 1790 1960 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:94, 1:4] 1 354 824 1308 2375 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:100, 1:4] 1 461 1790 1817 1975 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:46, 1:4] 1 328 1790 3562 5395 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:130, 1:4] 1 157 233 505 1913 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:120, 1:4] 1 122 181 510 1790 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:91, 1:4] 1 349 837 1790 3031 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
```

```
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:78, 1:4] 1 587 899 1579 1790 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:83, 1:4] 1 286 1108 1296 1418 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:69, 1:4] 1 439 1790 2466 3562 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:40, 1:4] 1 81 379 510 3562 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:82, 1:4] 1 436 487 3562 3804 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:61, 1:4] 1 433 990 1342 1790 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:63, 1:4] 1 1790 3031 3562 6749 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:73, 1:4] 1 584 1031 1620 1790 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:90, 1:4] 1 355 1219 1401 1790 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:28, 1:4] 1 3562 7811 11345 12213 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:78, 1:4] 1 437 1722 1790 1854 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:81, 1:4] 1 579 892 1289 1690 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:139, 1:4] 1 294 387 1468 1859 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:79, 1:4] 1 293 605 1763 1790 ...
```

```
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:82, 1:4] 1 223 409 511 1114 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:76, 1:4] 1 448 486 513 571 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:89, 1:4] 1 466 1790 2390 3031 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:76, 1:4] 1 1790 2221 3031 3562 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:137, 1:4] 1 155 256 330 1290 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
```

Load the probe locations

```
> load("mip1.mark.RData")
> head(mip1.mark)
```

```
      Chrom Chrom.Pos
19503     1   2201241
50000     1   2779554
35766     1   2905810
53523     1   3097040
56321     1   3117988
403       1   3128433
```

This is the raw copy number data.

To reconstruct this object, retrieve the cisplatin data from GEO, and load the copy numbers.

```
> load("dfci1.cna.RData")
> head(dfci1.cna)
```

```
      chr     pos       X1T      X3T      X4T      X5T      X6T      X7T
19503   1 2201241 0.7322896 3.033040 2.675703 2.814382 1.312860 1.705795
50000   1 2779554 1.0434465 2.467781 3.468592 1.852579 1.336153 1.554029
35766   1 2905810 0.7644918 2.635268 3.091216 1.919547 1.655401 1.968056
53523   1 3097040 0.4989782 2.732088 2.139864 2.219023 1.219798 1.478493
56321   1 3117988 0.7234751 2.360894 1.961882 1.681449 1.389884 1.507168
403     1 3128433 0.9007065 2.529160 3.024252 2.811464 2.188905 1.639370
           X8T       X9T     X10T     X11T     X12T     X13T     X14T     X15T
19503 1.063925 1.5399472 2.736378 1.610743 2.243210 1.987595 2.763606 1.763510
50000 1.660793 1.3794428 2.573883 1.365603 1.976099 2.008478 2.127013 2.198423
35766 1.290003 1.9922068 2.181665 2.200951 2.333128 1.536065 2.253574 2.034503
```

```
53523 1.026464 1.5037880 2.424278 1.521576 1.871073 1.801794 1.897132 1.656681
56321 1.155907 0.8941049 2.594061 1.483789 1.833122 1.910287 1.906536 1.576959
403   1.345216 2.2676301 2.302845 2.685832 2.165062 1.337980 2.584581 2.057897
         X16T     X17T     X18T     X20T      X21T     X22T     X23T     X24T
19503 2.050863 3.468619 1.779076 2.578868 1.7608849 2.884349 1.900527 1.571811
50000 1.677274 1.731927 1.714514 2.005048 0.5663026 1.604559 2.324549 1.970072
35766 1.960299 2.921513 1.744089 1.817376 0.8532232 1.921688 1.686386 1.790284
53523 1.839832 2.182616 2.111284 2.053501 0.3129118 2.043438 1.569914 1.519021
56321 1.626000 1.328629 1.486406 1.111131 1.4467178 1.293787 1.847027 1.752481
403   2.861951 4.635340 1.419152 2.348123 2.2801015 2.967830 1.784112 1.771836
         X25T     X26T     X27T     X28T     X29T
19503 7.734831 1.573455 4.602028 2.365204 1.633720
50000 1.723822 1.529922 2.452629 1.682278 1.267092
35766 2.940751 1.775539 3.604807 2.074486 1.465220
53523 5.541537 1.227875 2.705153 1.705511 1.537661
56321 1.389048 1.130569 2.982483 1.167176 1.256587
403   3.624598 3.028262 4.629879 1.986001 1.895491
```

This is the log2 transformed, centered, segmented copy number data.

Segmentation was performed using the R package DNAcopy with default parameters.

```
> load("dfci1.cna.seg.RData")
> head(dfci1.cna.seg)
```

```
   ID chrom loc.start  loc.end num.mark seg.mean
1 X1T     1   2201241  9449112      133  -1.1398
2 X1T     1   9821894 26905842      250   0.0519
3 X1T     1  27216671 31047379       52  -0.5478
4 X1T     1  31357051 53946601      230   0.3095
5 X1T     1  54234328 55212450       21   1.4243
6 X1T     1  55281461 61787323      130   0.3348
```

This is the patient response data using the Miller-Payne grade

```
> load("dfci1.mpgrade.RData")
> dfci1.mpgrade
```

```
 P1  P3  P4  P5  P6  P7  P8  P9 P10 P11 P12 P13 P14 P15 P16 P17 P18 P20 P21 P22
  3   5   1   5   1   4   4   5   4   3   1   1   2   0   1   5   5   2   0   2
P23 P24 P25 P26 P27 P28 P29
  3   2   3   0   1   2   5
```

This is the patient germline BRCA1/2 mutation status (1 means mutation)

```
> load("dfci1.brca.RData")
> dfci1.brca
```

```
 P1  P3  P4  P5  P6  P7  P8  P9 P10 P11 P12 P13 P14 P15 P16 P17 P18 P20 P21 P22
  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
P23 P24 P25 P26 P27 P28 P29
  0   1   0   0   0   0   0
```

Organize Cisplatin-1 trial data

First, organize the output from ASCAT, to get it into a more compact format

```
> dfci1.ascat.seg <- organize.ascat.segments(dfci1.ascat.output,
+     mip1.mark)
```

Now we call allelic imbalance based on the output from ASCAT. Any time the two alleles are not represented in equal amounts (balanced), that is defined as AI

```
> AI <- c(0, 1)[match(dfci1.ascat.seg[, 6] == dfci1.ascat.seg[,
+     7], c("TRUE", "FALSE"))]
> dfci1.ascat.seg <- cbind(dfci1.ascat.seg[, 1:5], AI, dfci1.ascat.seg[,
+     6:7])
```

Check if any samples failed ASCAT

```
> dfci1.ascat.output$failedarrays

character(0)
```

None failed

Now we call the number of telomeric, interstitial, and whole chromosome AI in each patient

```
> min.probes.cis1 <- 25
> no.tel.dfci1 <- no.tel(dfci1.ascat.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cis1)
> head(no.tel.dfci1)

    Telomeric AI Mean size Interstitial AI Mean Size Wholo chr AI 1 2 3 4 5 6 7
X1T           29  26742737              58  20015245            1 0 0 0 0 0 0 0
X3T           33  17937674              61  22787871            0 0 0 0 0 0 0 0
X4T           22  38135634              22  23873399            1 0 0 0 0 0 0 0
X5T           26  33778339              31  25402564            1 0 0 0 0 0 0 0
X6T           17  53082764               9  46576320            2 0 0 0 0 0 0 0
X7T           29  22273604              63  19399097            1 0 0 0 0 0 0 0
    8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
X1T 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
X3T 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
X4T 0 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
X5T 0 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
X6T 0 0  0  0  0  0  0  1  0  0  0  0  0  0  1  0
X7T 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1

> no.tel.cna.dfci1 <- no.tel.cna(dfci1.cna.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cis1)
> head(no.tel.cna.dfci1)

    Total telomeric CNA Mean size Telomeric gain Mean size Telomeric loss
X1T                  31  20594409             18  15397121             13
X3T                  18  13173028             18  13173028              0
X4T                  11  26511695              6  28641692              5
X5T                  14  16221799             13  14255329              1
X6T                   5  33009245              2  19790424              3
X7T                  22  10211666             21  10634565              1
    Mean size Total whole chromosome CNA Whole chromosome gain
X1T  27790653                         0                    0
X3T       NaN                         0                    0
X4T  23955699                         0                    0
X5T  41785913                         0                    0
X6T  41821792                         0                    0
X7T   1330786                         0                    0
```

```
    Whole chromosome loss
X1T                     0
X3T                     0
X4T                     0
X5T                     0
X6T                     0
X7T                     0
```

```
> no.cna.dfci1 <- gaps(dfci1.cna.seg, min.probes = min.probes.cis1)
> head(no.cna.dfci1)
```

```
    Short Aberrations Long Aberrations Short gains Short deletions Total Gains
X1T                 0              111           0                0          78
X3T                 0               95           0                0          94
X4T                 0               44           0                0          28
X5T                 0               39           0                0          37
X6T                 0               20           0                0           8
X7T                 0              105           0                0          99
    Total Deletions Total Aberrations
X1T              33               111
X3T               1                95
X4T              16                44
X5T               2                39
X6T              12                20
X7T               6               105
```

2.4 Load the Cisplatin-2 trial data

This is the output from ASCAT based on the log2 ratios and B-allele frequencies. To reconstruct this object, retrieve the cisplatin data from GEO, and calculate log2 ratios and BAF either using the submitted processed copy numbers and intensities, or by using the raw cel files.

```
> load("dfci2.ascat.output.RData")
> str(dfci2.ascat.output)

List of 7
 $ nA                  : num [1:329951, 1:45] 4 4 4 4 4 4 2 2 4 4 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:329951] "amp184660" "amp947" "amp53303" "amp364886" ...
  .. ..$ : chr [1:45] "P3" "P6" "P7" "P10" ...
 $ nB                  : num [1:329951, 1:45] 0 0 0 0 0 0 2 2 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:329951] "amp184660" "amp947" "amp53303" "amp364886" ...
  .. ..$ : chr [1:45] "X0721101_A01" "X0721101_A02" "X0721101_A03" "X0721101_A04" ...
 $ aberrantcellfraction: num [1:45] 0.59 0.71 0.57 0.29 0.68 0.34 0.81 0.32 0.49 0.68 ...
 $ ploidy              : num [1:45] 1.94 1.95 2.08 2.65 1.78 ...
 $ psi                 : num [1:45] 1.95 1.95 2.05 2.55 1.8 4.05 2.1 1.75 3 2.45 ...
 $ failedarrays        : chr [1:5] "P12" "P30" "P42" "P19" ...
 $ segments            :List of 50
  ..$ : num [1:286, 1:4] 1 525 715 1039 1065 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:202, 1:4] 1 877 3096 6995 7370 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:174, 1:4] 1 1298 5228 5264 5283 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:91, 1:4] 1 14775 14777 14815 15334 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : logi NA
  ..$ : num [1:124, 1:4] 1 680 3078 3414 11256 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:100, 1:4] 1 1507 1521 14509 14516 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:187, 1:4] 1 4353 5777 8526 10927 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:95, 1:4] 1 14509 15538 15689 17462 ...
  .. ..- attr(*, "dimnames")=List of 2
```

```
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:117, 1:4] 1 1133 1147 4191 5036 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:70, 1:4] 1 807 7874 8346 10390 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:154, 1:4] 1 2055 6378 16341 16514 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:114, 1:4] 1 6384 14748 14775 17806 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:94, 1:4] 1 1788 2353 3904 14509 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:142, 1:4] 1 1594 2295 3114 3178 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:159, 1:4] 1 489 1340 1397 1591 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:241, 1:4] 1 1917 6365 7915 9202 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:223, 1:4] 1 1687 1892 2595 3307 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:96, 1:4] 1 2391 2430 15597 15685 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:197, 1:4] 1 393 4039 6322 13835 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:185, 1:4] 1 2276 2373 2376 2430 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:101, 1:4] 1 2367 4209 4715 5175 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
```

```
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:164, 1:4] 1 1383 4013 4022 4080 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:95, 1:4] 1 1496 1552 1658 14509 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : logi NA
..$ : num [1:194, 1:4] 1 1003 1120 3011 8251 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:150, 1:4] 1 6378 14509 16702 21423 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:130, 1:4] 1 4619 8319 11227 11428 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:183, 1:4] 1 2151 2187 3993 6380 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:175, 1:4] 1 467 677 695 2441 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:32, 1:4] 1 27722 61145 82223 100190 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:163, 1:4] 1 118 1156 1252 10744 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:39, 1:4] 1 21940 21943 27722 41995 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:45, 1:4] 1 27722 58633 58718 61145 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:31, 1:4] 1 16455 16512 27722 33793 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
..$ : num [1:178, 1:4] 1 1051 1294 2815 2998 ...
.. ..- attr(*, "dimnames")=List of 2
```

```
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:156, 1:4] 1 3484 11289 11310 11641 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:157, 1:4] 1 4052 4065 5941 6352 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:138, 1:4] 1 14509 15750 21433 22092 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:48, 1:4] 1 14509 26483 27722 59390 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:173, 1:4] 1 3951 4026 4037 9379 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:76, 1:4] 1 2849 13820 13936 14509 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:27, 1:4] 1 21817 22137 27722 61145 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:27, 1:4] 1 27722 33237 33252 33268 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:34, 1:4] 1 27722 61015 61145 82223 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:54, 1:4] 1 21818 21956 22059 22074 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
  ..$ : num [1:142, 1:4] 1 142 15240 15694 19523 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:4] "start" "end" "nA" "nB"
```

Load the probe locations

```
> load("mip2.mark.RData")
> head(mip2.mark)
```

```
          Chr    Pos
amp184660   1  59369
amp947      1 524446
```

```
amp53303     1 552824
amp364886    1 683543
amp121615    1 744055
amp288732    1 744264
```

This is the raw copy number data.

To reconstruct this object, retrieve the cisplatin data from GEO, and load the copy numbers.

```
> load("dfci2.cna.RData")
> head(dfci2.cna)
```

```
          Chromosome Position     P3     P4     P5     P6     P7     P9    P10
amp184660          1    59369 1.6856 1.3543 1.0100 1.8666 1.5539 1.6091 1.7150
amp947             1   524446 2.4045 3.0214 6.2492 4.8599 6.3046 5.3291 2.0681
amp53303           1   552824 1.1540 0.9338 0.7626 0.9134 0.8943 1.3836 0.8250
amp364886          1   683543 1.3990 2.5582 2.1515 2.5753 1.7891 0.8916 2.3242
amp121615          1   744055 2.1820 1.4104 1.0626 1.4087 1.2543 2.1538 1.0602
amp288732          1   744264 4.9639 1.5209 1.9715 1.9310 1.9866 2.6213 1.8064
              P12    P13    P14    P15    P17    P18    P19    P20    P21
amp184660  0.3588 2.2730 1.2358 1.5687 2.0008 0.7247 1.3374 1.5673 0.6836
amp947    -0.0200 2.9073 5.1594 4.7980 2.0523 3.4927 3.4999 5.0331 2.9918
amp53303   0.3605 1.3682 0.7386 1.4765 0.6060 0.6779 0.7812 2.0723 0.4630
amp364886  2.4607 2.7326 2.1346 2.0244 2.2404 1.6447 2.2963 2.6832 1.6447
amp121615  0.5685 2.1658 0.9693 1.3474 1.1382 0.5628 0.8342 1.6142 0.6159
amp288732  2.2500 2.9689 2.2714 2.3334 1.3610 1.1392 1.3490 2.7497 0.9734
              P24    P26     P27    P28    P29    P30    P32    P33    P34
amp184660 0.8307 2.1412  1.5159 2.0271 1.5289 1.4844 1.4563 1.4315 0.8634
amp947    0.7731 4.6372 13.5112 2.1613 0.8438 5.0372 0.7705 5.3915 9.5061
amp53303  0.7081 1.3915  1.4553 0.9354 0.4733 0.6398 0.9077 1.2393 0.5607
amp364886 2.4672 1.6165  4.2043 2.9222 2.7542 2.5122 1.6025 2.5392 0.9930
amp121615 0.8136 1.4012  1.0481 0.6906 1.9269 0.7797 1.6878 0.6717 1.1294
amp288732 0.7855 2.0340  2.8002 1.2249 2.6238 2.2223 3.2103 0.8852 3.3948
              P35    P36    P37    P39    P40    P41    P42    P43    P46    P48
amp184660 1.2485 2.7688 1.4051 2.7214 1.9933 1.4166 0.8789 0.9012 1.9007 0.6528
amp947    4.0185 1.5600 2.1964 4.5751 4.3604 3.6977 4.0414 2.2604 5.5044 7.0190
amp53303  0.8630 1.5967 0.7664 1.6556 1.5178 0.8024 0.6633 0.6035 0.9527 1.0629
amp364886 2.6997 2.1458 3.3912 2.4377 2.4183 2.2254 2.1759 3.1786 1.6868 2.7243
amp121615 1.0521 1.4217 1.4085 1.6062 1.2727 1.0984 1.0815 1.1966 1.8054 0.7447
amp288732 2.9642 2.6145 1.8097 2.6413 2.5504 2.1996 1.7574 2.0518 2.0418 2.2442
              P49    P50    P51   P45L   P45R
amp184660 1.0853 1.2873 0.6256 1.4476 1.3291
amp947    3.8069 5.0468 3.5486 2.3083 3.1905
amp53303  0.8701 0.9387 0.4400 0.8018 0.5623
amp364886 3.4353 2.1732 2.3056 1.6513 2.1951
amp121615 1.4685 1.6974 0.6598 1.1061 0.7816
amp288732 3.6574 3.0803 1.5338 1.2814 2.0949
```

This is the log2 transformed, centered, segmented copy number data.

Segmentation was performed using the R package DNAcopy with default parameters.

```
> load("dfci2.cna.seg.RData")
> head(dfci2.cna.seg)
```

```
  ID chrom loc.start  loc.end num.mark seg.mean
1 P3     1     59369   788822        8   0.0135
2 P3     1   1038818  3749644      115   0.9012
3 P3     1   3751363  6052451      290   0.3382
4 P3     1   6062727  6646393       58   0.8899
5 P3     1   6662360  9930808      346   0.2469
6 P3     1   9935601 11755953      156  -0.0810
```

This is the patient response data using the Miller-Payne grade

```
> load("dfci2.mpgrade.RData")
> dfci2.mpgrade
```

```
  P3   P4   P5   P6   P7   P9  P10  P12  P13  P14  P15  P17  P18  P19  P20  P21
   4    3    3    4    5    1    1    1    3    1    5    5    1    2    2    2
 P24  P26  P27  P28  P29  P30  P32  P33  P34  P35  P36  P37  P39  P40  P41  P42
   2    4    4    0    1    1    1   NA    1    1    5    3    2    4    5    2
 P43  P46  P48  P49  P50  P51 P45L P45R
  NA    1    4    1   NA    5    1    5
```

This is the patient germline BRCA1/2 mutation status (1 means mutation)

```
> load("dfci2.brca.RData")
> dfci2.brca
```

```
  P3   P4   P5   P6   P7   P9  P10  P12  P13  P14  P15  P17  P18  P19  P20  P21
   0    0    0    0    1    0    0    0    0    0    1    0    0    0    1    0
 P24  P26  P27  P28  P29  P30  P32  P33  P34  P35  P36  P37  P39  P40  P41  P42
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
 P43  P46  P48  P49  P50  P51 P45L P45R
   0    0    0    0    0    0    0    0
```

Organize the Cisplatin-2 trial data

First, organize the output from ASCAT, to get it into a more compact format

```
> dfci2.ascat.seg <- organize.ascat.segments(dfci2.ascat.output,
+     mip2.mark)
```

Now we call allelic imbalance based on the output from ASCAT. Any time the two alleles are not represented in equal amounts (balanced), that is defined as AI

```
> AI <- c(0, 1)[match(dfci2.ascat.seg[, 6] == dfci2.ascat.seg[,
+     7], c("TRUE", "FALSE"))]
> dfci2.ascat.seg <- cbind(dfci2.ascat.seg[, 1:5], AI, dfci2.ascat.seg[,
+     6:7])
```

Check if any samples failed ASCAT

```
> dfci2.ascat.output$failedarrays
```

```
[1] "P12" "P30" "P42" "P19" "P36"
```

5 failed

Remove samples where ASCAT failed, samples from patients that went off trial, and normal samples.

```
> dfci2.samples <- intersect(names(dfci2.mpgrade[!is.na(dfci2.mpgrade)]),
+     unique(dfci2.ascat.seg[, 1]))
```

```
> dfci2.mpgrade <- dfci2.mpgrade[dfci2.samples]
> dfci2.brca <- dfci2.brca[dfci2.samples]

> min.probes.cis2 <- 200
> no.tel.dfci2 <- no.tel(dfci2.ascat.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cis2)[dfci2.samples, ]
> head(no.tel.dfci2)

   Telomeric AI Mean size Interstitial AI Mean Size Wholo chr AI 1 2 3 4 5 6 7
P3           33   9621078             135  14881010            0 0 0 0 0 0 0 0
P4           22  17062383              82  17221975            0 0 0 0 0 0 0 0
P5           12  31685950              23  21686632            5 0 0 0 1 0 0 0
P6           30  22204775              79  13743492            0 0 0 0 0 0 0 0
P7           23  16864291              64  15306967            0 0 0 0 0 0 0 0
P9           25  28508299              61  18168760            2 0 0 0 0 0 0 0
   8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
P3 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
P4 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
P5 0 0  0  0  0  1  0  0  0  0  0  0  1  0  1  1
P6 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
P7 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
P9 0 0  0  0  0  1  0  0  0  0  0  0  0  0  1  0

> no.tel.cna.dfci2 <- no.tel.cna(dfci2.cna.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cis2)[dfci2.samples, ]
> head(no.tel.cna.dfci2)

   Total telomeric CNA Mean size Telomeric gain Mean size Telomeric loss
P3                  14   5505118             12   4220657              2
P4                   8   6536302              7   4849746              1
P5                   8   9891869              8   9891869              0
P6                  19  13167745             13   6246089              6
P7                  20  15009410             13  10556676              7
P9                  20  19590463              3   5969351             17
   Mean size Total whole chromosome CNA Whole chromosome gain
P3  13211880                          0                     0
P4  18342191                          1                     1
P5       NaN                          1                     1
P6  28164666                          2                     2
P7  23278773                          0                     0
P9  21994188                          0                     0
   Whole chromosome loss
P3                     0
P4                     0
P5                     0
P6                     0
P7                     0
P9                     0

> no.cna.dfci2 <- gaps(dfci2.cna.seg, min.probes = min.probes.cis2)[dfci2.samples,
+     ]
> head(no.cna.dfci2)

   Short Aberrations Long Aberrations Short gains Short deletions Total Gains
P3                 0              109           0               0          89
```

P4	0	73	0	0	34
P5	0	42	0	0	40
P6	0	123	0	0	71
P7	0	78	0	0	49
P9	0	111	0	0	24

	Total Deletions	Total Aberrations
P3	20	109
P4	39	73
P5	2	42
P6	52	123
P7	29	78
P9	87	111

2.5 Load TCGA ovarian data

This is the segmented output from ASCAT based on the log2 ratios and B-allele frequencies. As the ASCAT output for the TCGA ovarian cohort is very big, I am here loading an object where I have already run the "organize.ascat.segments" function. To reconstruct this object, retrieve the TCGA data from the TCGA website, calculate log2 ratios and BAF using your prefferred method, then process it using the above function to organize the segments.

```
> load("tcga_ovarian_ascat_seg_sw.RData")
> head(tcga_ovarian_ascat_seg_sw)
```

```
      SampleID Chr    Start      End nProbes nA nB
1 TCGA-13-1408   1    51599 26929952   15349  3  1
2 TCGA-13-1408   1 26940267 46034716   10842  2  0
3 TCGA-13-1408   1 46034769 47656816     867  2  2
4 TCGA-13-1408   1 47657169 49221330    1189  3  1
5 TCGA-13-1408   1 49222792 60057280    7180  2  2
6 TCGA-13-1408   1 60057975 62057956    1442  2  1
```

This is the ASCAT based contamination estimates for each TCGA sample

```
> load("tcga_ovarian_ascat_aberrant.20111220.RData")
> head(tcga_ovarian_ascat_aberrant)
```

```
TCGA-13-1408 TCGA-13-1409 TCGA-13-1477 TCGA-13-1482 TCGA-13-1483 TCGA-13-1484
        0.42         0.93         0.84         0.84         0.73         0.45
```

This is the patient response data acquired from the TCGA website, clinical info.

Outcome, followup, and recurrence are from the clinical_patient file, and based on the headers: primary_therapy_outcome_success, days_to_last_followup and days_to_tumor_recurrence. Platinum drug status is from the file clinical_drug, and the column drug_name

```
> load("tcga.ovarian.recurrence.20111220.RData")
> head(tcga.ovarian.recurrence)
```

```
TCGA-13-1408 TCGA-13-1409 TCGA-13-1477 TCGA-13-1482 TCGA-13-1483 TCGA-13-1484
      "null"       "null"       "null"        "602"        "296"        "496"
```

```
> load("tcga.ovarian.followup.20111220.RData")
> head(tcga.ovarian.followup)
```

```
TCGA-13-1408 TCGA-13-1409 TCGA-13-1477 TCGA-13-1482 TCGA-13-1483 TCGA-13-1484
         180          208         1661         1877          894         2982
```

```
> load("tcga.ovarian.outcome.20111220.RData")
> head(tcga.ovarian.outcome)
```

```
         TCGA-13-1408          TCGA-13-1409          TCGA-13-1477
   "PARTIAL RESPONSE"                "null" "PROGRESSIVE DISEASE"
         TCGA-13-1482          TCGA-13-1483          TCGA-13-1484
   "PARTIAL RESPONSE"    "PARTIAL RESPONSE"   "COMPLETE RESPONSE"
```

```
> load("tcga.ovarian.platinum.drug.20111220.RData")
> head(tcga.ovarian.platinum.drug)
```

```
TCGA-13-1408 TCGA-13-1409 TCGA-13-1477 TCGA-13-1482 TCGA-13-1483 TCGA-13-1484
        TRUE         TRUE         TRUE         TRUE         TRUE         TRUE
```

This is the BRCA1/2 mutation data, acquired from cBio website

```
> tcga.ov.brca.mut <- read.table("tcga.ov.brcamut.20120201.txt",
+     sep = "\t", header = T, row.names = 1)
> head(tcga.ov.brca.mut)

             BRCA1 BRCA2
TCGA-04-1331        MUT;
TCGA-04-1332
TCGA-04-1336        MUT;
TCGA-04-1337
TCGA-04-1338
TCGA-04-1342
```

This is the TCGA ovarian expression data. This data was downloaded as CEL files, and normalized using RMA as implemented in the AFFY package

```
> load("tcga.ovarian.20111206.rma.RData")
> tcga.ovarian.20111206.rma

ExpressionSet (storageMode: lockedEnvironment)
assayData: 22277 features, 584 samples
  element names: exprs
protocolData
  sampleNames:
    AGARS_p_TCGA_B12_RNA_ReDo_HT_HG-U133A_96-HTA_A01_443066.CEL
    AGARS_p_TCGA_B12_RNA_ReDo_HT_HG-U133A_96-HTA_A02_443064.CEL ...
    TARRE_p_MultiPlate_TCGA_SS_MA_Ref_HT_HG-U133A_96-HTA_F08_586108.CEL
    (584 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames:
    AGARS_p_TCGA_B12_RNA_ReDo_HT_HG-U133A_96-HTA_A01_443066.CEL
    AGARS_p_TCGA_B12_RNA_ReDo_HT_HG-U133A_96-HTA_A02_443064.CEL ...
    TARRE_p_MultiPlate_TCGA_SS_MA_Ref_HT_HG-U133A_96-HTA_F08_586108.CEL
    (584 total)
  varLabels: SampleID TissueID ... adjuvantCetuximab (51 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hthgu133a
```

Determine which TCGA ovarian samples are useful We are interested in samples with at least 6 months followup, platinum treated. We will define sensitive as no recurrence within 6 months, and either partial or complete response to therapy. Resistant will be defined as recurrence within 6 months, or stable or progressive disease

```
> tmp <- c(0, 1)[match(tcga.ovarian.recurrence == "null", c("TRUE",
+     "FALSE"))]
> tmp1 <- tcga.ovarian.followup
> tmp1[tmp == 1] <- as.numeric(tcga.ovarian.recurrence[tmp == 1])
> tmp <- tmp == 1 & tmp1 <= 180
> names(tmp) <- names(tcga.ovarian.recurrence)
> tcga.ov.outcome.180 <- tcga.ovarian.outcome %in% c("PROGRESSIVE DISEASE",
```

```
+     "STABLE DISEASE")
> tcga.ov.outcome.180[tmp] <- tmp[tmp]
> names(tcga.ov.outcome.180) <- names(tcga.ovarian.recurrence)
> tcga.ov.outcome.180 <- !tcga.ov.outcome.180
```

Now we call allelic imbalance based on the output from ASCAT. Any time the two alleles are not represented in equal amounts (balanced), that is defined as AI

```
> AI <- c(0, 1)[match(tcga_ovarian_ascat_seg_sw[, 6] == tcga_ovarian_ascat_seg_sw[,
+     7], c("TRUE", "FALSE"))]
> tcga_ovarian_ascat_seg <- cbind(tcga_ovarian_ascat_seg_sw[, 1:5],
+     AI, tcga_ovarian_ascat_seg_sw[, 6:7])
```

Now we call the number of telomeric, interstitial, and whole chromosome AI in each patient

```
> min.probes.tcga <- 500
> no.tel.tcga <- no.tel(tcga_ovarian_ascat_seg, chrominfo = chrominfo,
+     min.probes = min.probes.tcga)
> head(no.tel.tcga)
```

```
             Telomeric AI Mean size Interstitial AI Mean Size Wholo chr AI 1 2
TCGA-13-1408           28  15655312             125  11853485            0 0 0
TCGA-13-1409           19  51475372              41  15505762            4 0 0
TCGA-13-1477            6  56094620               4   7314540            1 0 0
TCGA-13-1482           24  21894087              38  18546167            1 0 0
TCGA-13-1483           24  24978548              37  15408026            1 0 0
TCGA-13-1484           11  46165898              17  20799284            5 0 0
             3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
TCGA-13-1408 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
TCGA-13-1409 0 0 0 0 0 0 0  0  1  1  0  0  0  0  0  1  0  0  0  1  0
TCGA-13-1477 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
TCGA-13-1482 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
TCGA-13-1483 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
TCGA-13-1484 0 0 0 0 1 0 0  0  1  0  0  0  0  1  1  0  0  0  0  1  0
```

Organize the expression data

```
> tcga.ovarian.eset <- exprs(tcga.ovarian.20111206.rma)
> colnames(tcga.ovarian.eset) <- tcga.ovarian.20111206.rma$SampleID
```

As we are going to use the expression data to investigate BRCA1 expression, we will now identify the optimum probe set to use. Affy identifies two as targeting BRCA1: 204531_s_at and 211851_x_at. To find out which is better, we use the R package JetSet to find the one that matches better the BRCA1 gene, and is most resistant to degradation.

```
> brca.probe <- jmap(chip = "hgu133a", symbol = "BRCA1")
> brca.probe
```

```
        BRCA1
"204531_s_at"
```

2.6 Load TCGA breast data

As the TCGA breast data is based on patients that received standard therapy which usually does not include platinum, we only used it to test how the correlation is between BRCA1 expression and NtAI, restricted to the triple negative subtype to match the cisplatin trials sample populations

This is the gene expression data, for the TCGA breast cohort as it was on january 5th, 2012. It is normalized and summarized by TCGA (level 3)

```
> load("tcga.breast.ge.level3.RData")
```

Now we subtype the samples to get a TNBC only cohort

```
> tmp <- pam(cbind(scale(tcga.breast.ge.level3["ESR1", ]), scale(tcga.breast.ge.level3["ERBB2",
+     ])), 3)$cluster
> tcga.breast.pam.sub <- c("ER", "HER2", "TNBC")[match(tmp, c(1,
+     2, 3))]
> names(tcga.breast.pam.sub) <- names(tmp)
> head(tcga.breast.pam.sub)

TCGA-A1-A0SD-01 TCGA-A1-A0SE-01 TCGA-A1-A0SH-01 TCGA-A1-A0SJ-01 TCGA-A1-A0SK-01
           "ER"            "ER"          "HER2"            "ER"          "TNBC"
TCGA-A1-A0SM-01
         "HER2"

> table(tcga.breast.pam.sub)

tcga.breast.pam.sub
  ER HER2 TNBC
 371   59  106
```

This gives us 106 samples TNBC. Of these, there were at the 5th of January 2012 only SNP data with matched normal for 102, which we download, pre-process, and run through ASCAT in the same way as for the TCGA ovarian cancer cohort

This is the ASCAT based contamination estimates for each TCGA sample

```
> load("tcga_breast_tnbc.cont.sw.RData")
> head(tcga_ovarian_ascat_aberrant)

TCGA-13-1408 TCGA-13-1409 TCGA-13-1477 TCGA-13-1482 TCGA-13-1483 TCGA-13-1484
        0.42         0.93         0.84         0.84         0.73         0.45
```

This is the ASCAT based contamination estimates for each TCGA sample

```
> load("tcga_breast_tnbc.segments.sw.RData")
> head(tcga_ovarian_ascat_seg_sw)

      SampleID Chr    Start      End nProbes nA nB
1 TCGA-13-1408   1    51599 26929952   15349  3  1
2 TCGA-13-1408   1 26940267 46034716   10842  2  0
3 TCGA-13-1408   1 46034769 47656816     867  2  2
4 TCGA-13-1408   1 47657169 49221330    1189  3  1
5 TCGA-13-1408   1 49222792 60057280    7180  2  2
6 TCGA-13-1408   1 60057975 62057956    1442  2  1
```

Process and organize the TNBC data

```
> AI <- c(0, 1)[match(tcga_breast_tnbc.segments.sw[, 6] == tcga_breast_tnbc.segments.sw[,
+     7], c("TRUE", "FALSE"))]
> tcga_breast_tnbc.segments.sw <- cbind(tcga_breast_tnbc.segments.sw[,
+     1:5], AI, tcga_breast_tnbc.segments.sw[, 6:7])
> min.probes.tcga <- 500
> no.tel.tcga.tnbc <- no.tel(tcga_breast_tnbc.segments.sw, chrominfo = chrominfo,
+     min.probes = min.probes.tcga, min.size = 0)
> head(no.tel.tcga.tnbc)
```

```
             Telomeric AI Mean size Interstitial AI Mean Size Wholo chr AI 1 2
TCGA-BH-A0B3           34  28646348              67  21267352            4 0 0
TCGA-BH-A0B9           27  22642919              76  18941498            0 0 0
TCGA-B6-A0IK           12  44507748              40  17153564            8 0 1
TCGA-BH-A0E0           21  40731889              37  27564557            9 0 0
TCGA-B6-A0RE           32  30806494              67  26488870            1 0 0
TCGA-AN-A04D           17  22293486              57  31195386            4 0 0
             3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
TCGA-BH-A0B3 0 0 0 0 0 0 1  0  0  0  1  1  0  0  0  0  0  0  0  1  0
TCGA-BH-A0B9 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
TCGA-B6-A0IK 0 0 0 0 1 0 0  0  0  1  1  1  0  1  0  0  1  0  0  1  0
TCGA-BH-A0E0 0 1 1 0 1 0 0  0  0  0  0  1  0  0  1  1  0  0  1  1  1
TCGA-B6-A0RE 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
TCGA-AN-A04D 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  1  0  0  1  1  0
```

Subset the expression data to match the SNP data

```
> colnames(tcga.breast.ge.level3) <- sub("-01$", "", colnames(tcga.breast.ge.level3))
> tcga.breast.ge.level3 <- tcga.breast.ge.level3[, rownames(no.tel.tcga.tnbc)]
> dim(tcga.breast.ge.level3)
```

```
[1] 17811    90
```

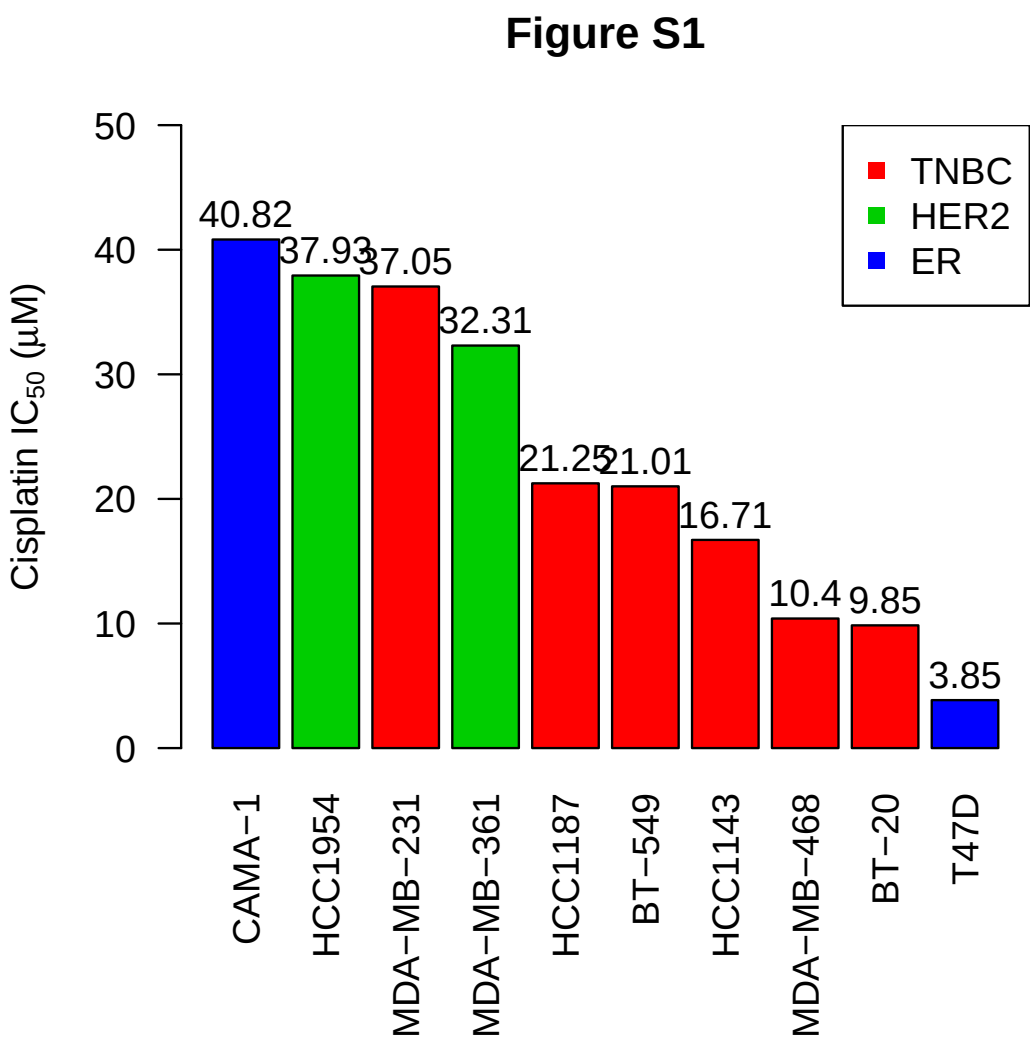
Unfortunately, ASCAT failed on 12 samples, leaving 90 with matched GE.

3 Cell lines

3.1 Figure 1

Plot figure 1a

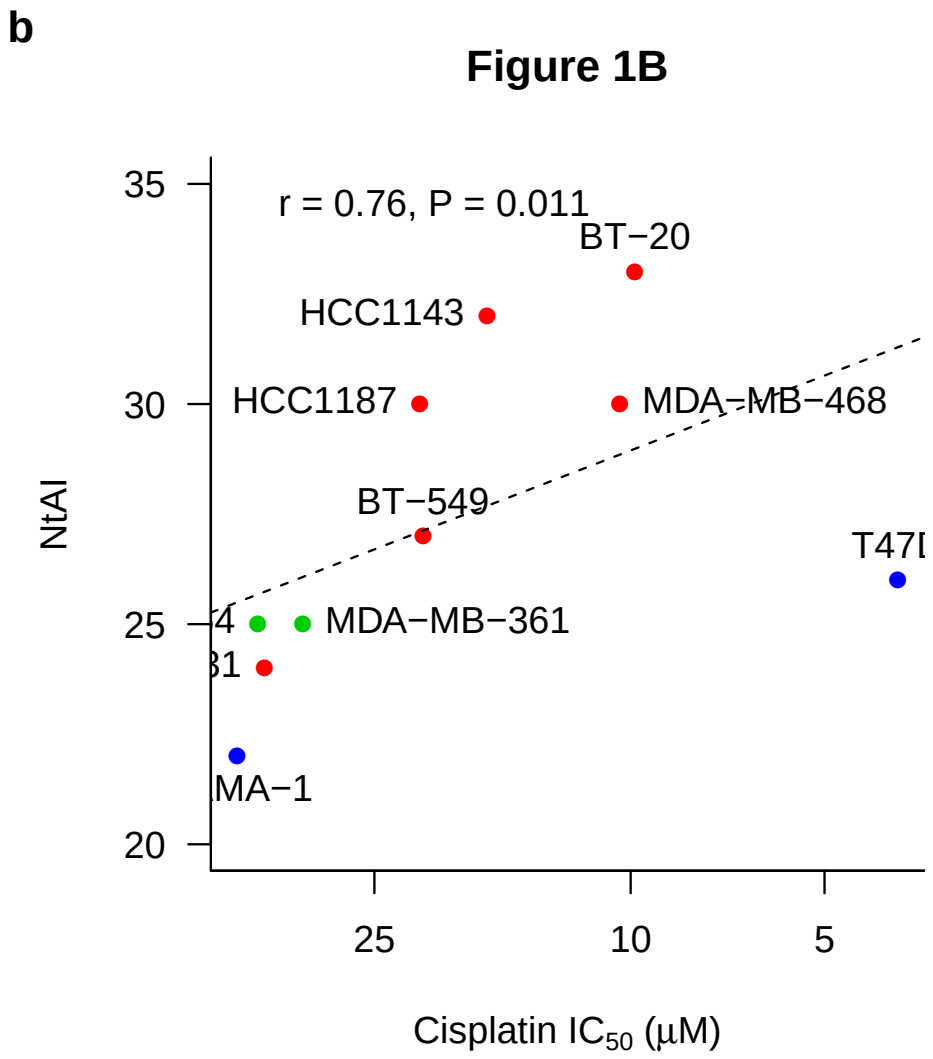
```
> par(las = 2, oma = c(2, 0, 0, 0))
> tmp <- barplot(sort(ic50.CL[, 1], decreasing = T), col = c(2,
+     3, 4)[match(ic50.CL[order(ic50.CL[, 1], decreasing = T),
+     2], c("TNBC", "HER2", "ER"))], names.arg = rownames(ic50.CL[order(ic50.CL[,
+     1], decreasing = T), ]), ylab = expression(paste("Cisplatin ",
+     IC[50], " (", mu, "M)")), main = "Figure S1", ylim = c(0,
+     50))
> text(tmp[, 1], sort(ic50.CL[, 1], decreasing = T) + 2, sort(ic50.CL[,
+     1], decreasing = T))
> legend("topright", legend = c("TNBC", "HER2", "ER"), col = c(2,
+     3, 4), pch = 15)
```



Plot figure 1b

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> subtype.col <- c(2, 3, 4)[match(ic50.CL[indx.cl, 2], c("TNBC",
+     "HER2", "ER"))]
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.CL[indx.cl, 1],
+     pch = 16, col = subtype.col, main = "Figure 1B", xlab = expression(paste("Cisplatin ",
+         IC[50], " (", mu, "M)")), ylab = "NtAI", ylim = c(20,
+         35), axes = F)
> pos.vec <- c(3, 3, 1, 2, 2, 2, 2, 4, 4, 3)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.CL[indx.cl, 1],
+     indx.cl, pos = pos.vec)
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> a <- lm(no.tel.CL[indx.cl, 1] ~ tmp)
> abline(a, lty = 2)
> a <- cor.test(no.tel.CL[indx.cl, 1], tmp, method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("b", xoff = 2)
```

b



Plot figure 1c

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> heiser.subtype.col <- c(2, 3, 4, 6)[match(heiser.cl.sub[indx.heiser],
+     c("Basal", "ERBB2AMP", "Luminal", "Claudin-low"))]
> plot(heiser.cl.gi50[indx.heiser, "Cisplatin"], no.tel.heiser.cl[indx.heiser,
+     1], pch = 16, col = heiser.subtype.col, main = "Figure 1C",
+     xlab = expression(paste("Cisplatin ", IC[50], " (", mu, "M)")),
+     ylab = "NtAI", ylim = c(0, 43), xlim = c(3, 6.5), axes = F)
> text(heiser.cl.gi50[indx.heiser, "Cisplatin"], no.tel.heiser.cl[indx.heiser,
+     1], indx.heiser, pos = 3)
> a <- lm(no.tel.heiser.cl[indx.heiser, 1] ~ heiser.cl.gi50[indx.heiser,
+     "Cisplatin"])
> abline(a, lty = 2)
> a <- cor.test(no.tel.heiser.cl[indx.heiser, 1], heiser.cl.gi50[indx.heiser,
+     "Cisplatin"], method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
```

```
> box()
> axis(2)
> axis(1, at = seq(3, 6, by = 1), tick = T, line = 0, labels = 10^-seq(3,
+     6, by = 1))
> label.panel("c", xoff = 2)
```

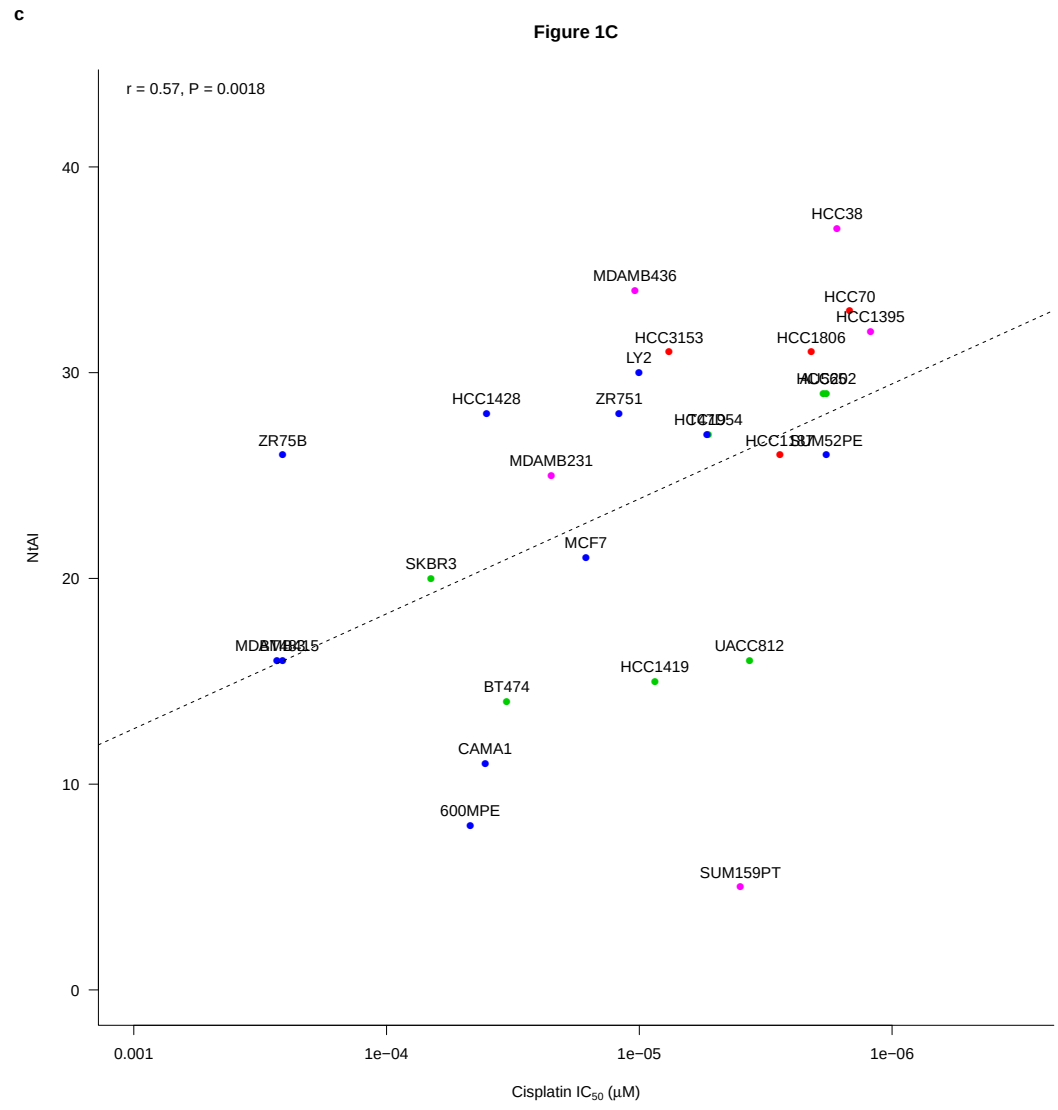


Figure 1C

Show which drug is the most strongly associated with NtAI

```
> heiser.r.drugs <- cbind(apply(heiser.cl.gi50[indx.heiser, ],
+     2, function(x) {
+         cor.test(x, no.tel.heiser.cl[indx.heiser, 1], method = "spearman")$estimate
+     }), apply(heiser.cl.gi50[indx.heiser, ], 2, function(x) {
+     cor.test(x, no.tel.heiser.cl[indx.heiser, 1], method = "spearman")$p.value
+ }))
> colnames(heiser.r.drugs) <- c("r", "p-value")
> heiser.r.drugs[order(heiser.r.drugs[, 1], decreasing = T), ]
```

```
                                r     p-value
Cisplatin                0.571560595 0.001843515
```

ZM.447439	0.425549091	0.061394925
GSK1070916	0.385817016	0.056802882
GSK461364	0.379650111	0.061227825
CPT.11	0.371639639	0.061568444
Carboplatin	0.355963902	0.068404000
GSK923295	0.346888426	0.089345534
TCS.2312.dihydrochloride	0.342583800	0.086673339
Topotecan	0.329509062	0.100213584
CGC.11047	0.328746731	0.094073123
VX.680	0.307279390	0.164205830
Etoposide	0.297603176	0.139797062
Vinorelbine	0.267890359	0.176702620
Glycyl.H.1152	0.263823659	0.223836459
Ixabepilone	0.210027760	0.293037817
Ibandronate.sodium.salt	0.209118863	0.338256651
NSC.663284	0.190891810	0.371585048
Docetaxel	0.185648557	0.363879159
PD173074	0.176645420	0.420076425
Ispinesib	0.172730254	0.388926792
Paclitaxel	0.160986756	0.432073729
Methotrexate	0.126782366	0.564308578
Nutlin.3a	0.116244187	0.555817585
Oxaliplatin	0.092660706	0.645740638
Pemetrexed	0.090689773	0.695832724
Bortezomib	0.088279784	0.655085157
AZD6244	0.087326792	0.722224757
AG1024	0.081836030	0.697361194
Erlotinib	0.078070347	0.698710648
Gemcitabine	0.072945274	0.723242362
Bosutinib	0.071864221	0.738606822
TPCA.1	0.070070992	0.756668622
CGC.11144	0.069092102	0.732023808
X5.FU	0.055954583	0.799818312
Triciribine	0.054723390	0.786314916
Lestaurtinib	0.044059492	0.841781863
BEZ235	0.027924727	0.906967664
MLN4924	0.018740520	0.930738589
NU6102	0.018504290	0.930042058
Sunitinib.Malate	0.011922973	0.952933032
Doxorubicin	0.011303325	0.956295526
Epirubicin	0.001743304	0.993549599
Tamoxifen	0.001546198	0.994147433
TCS.JNK.5a	0.000000000	1.000000000
GSK1120212	-0.004613722	0.981778494
Sorafenib	-0.025000037	0.903512912
Purvalanol.A	-0.045193979	0.833908296
PD.98059	-0.046711009	0.844959879
TGX.221	-0.066243080	0.753062253
AS.252424	-0.078125265	0.736412738
X5.FdUR	-0.084360655	0.688475702
Fascaplysin	-0.087435137	0.664535090
ICRF.193	-0.092608707	0.681878629
L.779450	-0.103980563	0.636821587
XRP44X	-0.125082076	0.560326102

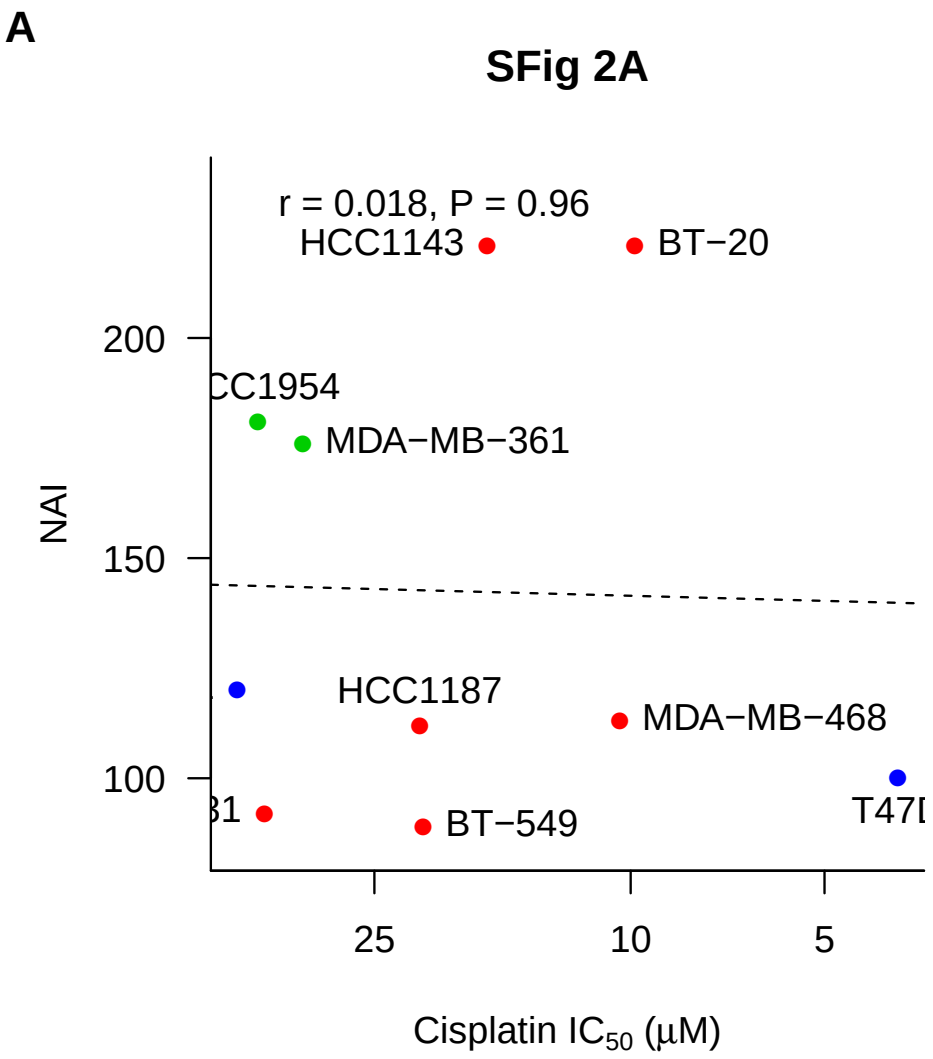
```
GSK1487371               -0.131333571 0.560173230
Lapatinib                -0.231616679 0.265267399
Vorinostat               -0.234133340 0.230454647
Oxamflatin               -0.247109373 0.233704676
GSK1059615               -0.258089675 0.212905196
X17.AAG                  -0.259158066 0.201095412
Geldanamycin             -0.259976485 0.199625554
Gefitinib                -0.260787916 0.198175511
AG1478                   -0.262569259 0.226128026
SB.3CT                   -0.272236509 0.187998085
BIBW2992                 -0.310620539 0.224950192
Trichostatin.A           -0.332557074 0.083791096
GSK2119563               -0.359829219 0.065247596
GSK1838705               -0.363537246 0.057223469
GSK2126458               -0.370835040 0.056875269
LBH589                   -0.374810403 0.071136525
Rapamycin                -0.379355976 0.067508033
Temsirolimus             -0.381295770 0.054613389
Sigma.AKT1.2.inhibitor   -0.434518572 0.033857624
```

NtAI is most strongly and most significantly correlated with Cisplatin

3.2 Supplementary figure 2

Plot supplementary figure 2a

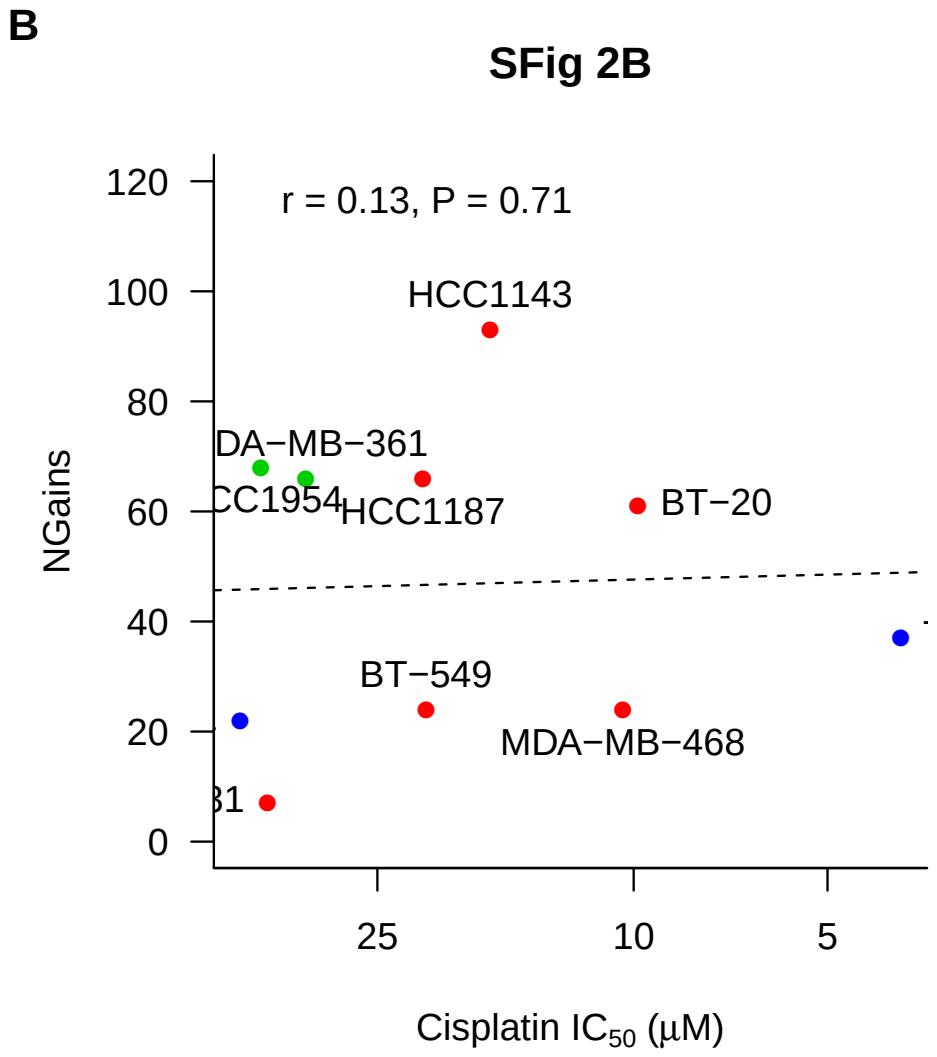
```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> a <- no.tel.CL[indx.cl, 1] + no.tel.CL[indx.cl, 3]
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), a, pch = 16, col = subtype.col,
+     main = "SFig 2A", xlab = expression(paste("Cisplatin ", IC[50],
+         " (", mu, "M)")), ylab = "NAI", ylim = c(85, 235), axes = F)
> pos.vec <- c(4, 4, 2, 2, 3, 3, 2, 4, 4, 1)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), a, indx.cl, pos = pos.vec,
+     )
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> b <- lm(a ~ tmp)
> abline(b, lty = 2)
> legend("topleft", legend = paste("r = ", signif(cor.test(a, tmp,
+     method = "spearman")$estimate, digits = 2), ", P = ", signif(cor.test(a,
+     tmp, method = "spearman")$p.value, digits = 2), sep = ""),
+     lty = -1, pch = -1, bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("A", xoff = 2)
```



Plot supplementary figure 2b

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> a <- no.cna.CL[indx.cl, 5] - no.cna.CL[indx.cl, 3]
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), a, pch = 16, col = subtype.col,
+     main = "SFig 2B", xlab = expression(paste("Cisplatin ", IC[50],
+         " (", mu, "M)")), ylab = "NGains", ylim = c(0, 120),
+     axes = F)
> pos.vec <- c(4, 3, 2, 3, 1, 1, 2, 3, 1, 4)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), a, indx.cl, pos = pos.vec,
+     )
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> b <- lm(a ~ tmp)
> abline(b, lty = 2)
> legend("topleft", legend = paste("r = ", signif(cor.test(a, tmp,
+     method = "spearman")$estimate, digits = 2), ", P = ", signif(cor.test(a,
+     tmp, method = "spearman")$p.value, digits = 2), sep = ""),
+     lty = -1, pch = -1, bty = "n")
> box()
```

```
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("B", xoff = 2)
```

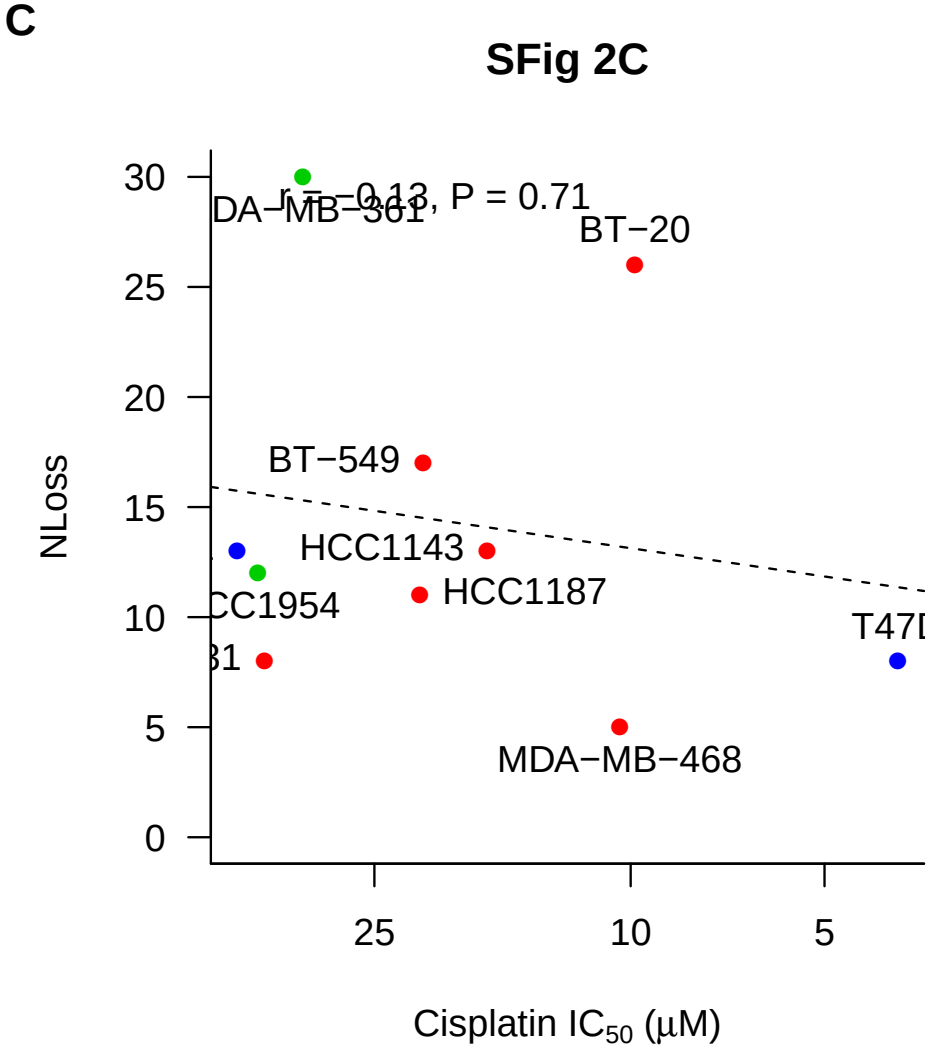


Plot supplementary figure 2c

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> a <- no.cna.CL[indx.cl, 6] - no.cna.CL[indx.cl, 4]
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), a, pch = 16, col = subtype.col,
+     main = "SFig 2C", xlab = expression(paste("Cisplatin ", IC[50],
+         " (", mu, "M)")), ylab = "NLoss", ylim = c(0, 30), axes = F)
> pos.vec <- c(3, 2, 2, 2, 4, 1, 2, 1, 1, 3)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), a, indx.cl, pos = pos.vec,
+     )
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> b <- lm(a ~ tmp)
> abline(b, lty = 2)
> legend("topleft", legend = paste("r = ", signif(cor.test(a, tmp,
```

```
+     method = "spearman")$estimate, digits = 2), ", P = ", signif(cor.test(a,
+     tmp, method = "spearman")$p.value, digits = 2), sep = ""),
+     lty = -1, pch = -1, bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("C", xoff = 2)
```

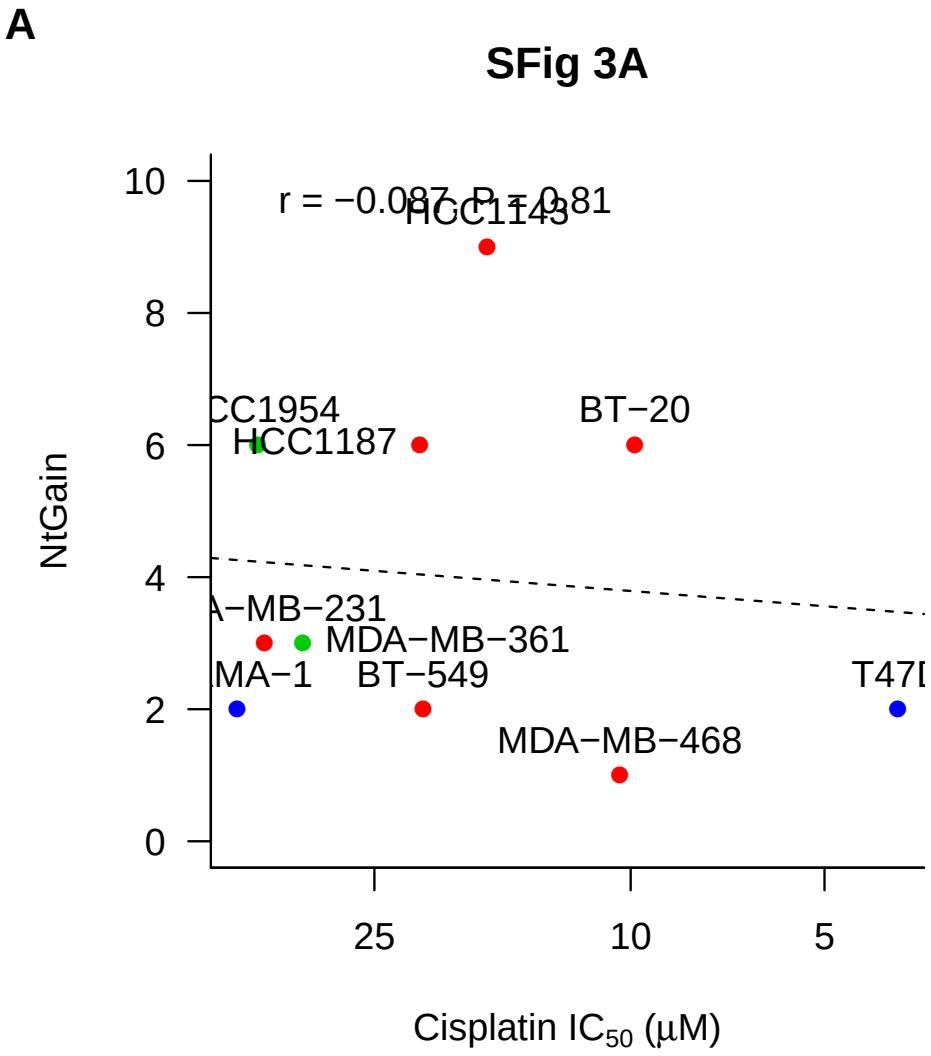
C



3.3 Supplementary figure 3

Plot supplementary figure 3A

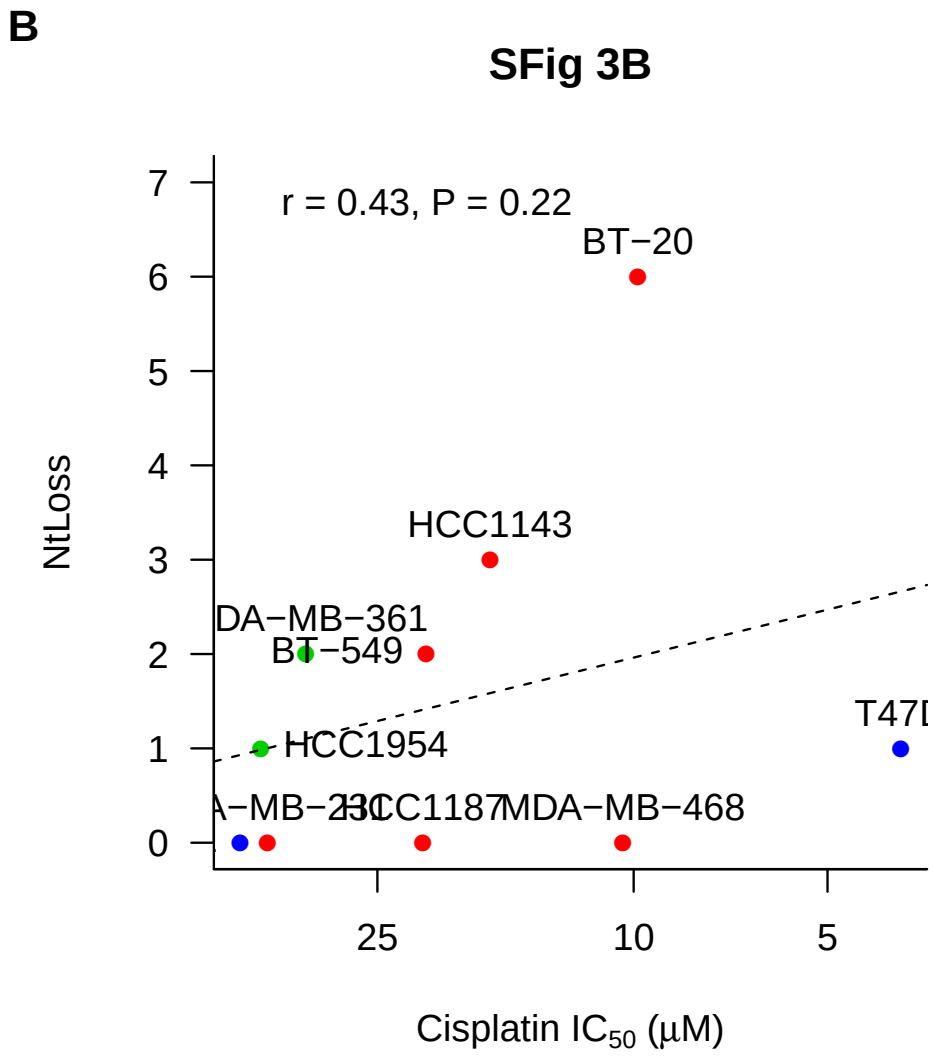
```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.cna.CL[indx.cl,
+     3], pch = 16, col = subtype.col, main = "SFig 3A", xlab = expression(paste("Cisplatin ",
+     IC[50], " (", mu, "M)")), ylab = "NtGain", ylim = c(0, 10),
+     axes = F)
> pos.vec <- c(3, 3, 3, 3, 2, 3, 3, 4, 3, 3)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.cna.CL[indx.cl,
+     3], indx.cl, pos = pos.vec)
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> a <- lm(no.tel.cna.CL[indx.cl, 3] ~ tmp)
> abline(a, lty = 2)
> a <- cor.test(no.tel.cna.CL[indx.cl, 3], tmp, method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("A", xoff = 2)
```



Plot supplementary figure 3B

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.cna.CL[indx.cl,
+     5], pch = 16, col = subtype.col, main = "SFig 3B", xlab = expression(paste("Cisplatin ",
+     IC[50], " (", mu, "M)")), ylab = "NtLoss", ylim = c(0, 7),
+     axes = F)
> pos.vec <- c(3, 2, 2, 3, 3, 4, 3, 3, 3, 3)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.cna.CL[indx.cl,
+     5], indx.cl, pos = pos.vec)
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> a <- lm(no.tel.cna.CL[indx.cl, 5] ~ tmp)
> abline(a, lty = 2)
> a <- cor.test(no.tel.cna.CL[indx.cl, 5], tmp, method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> box()
> axis(2)
```

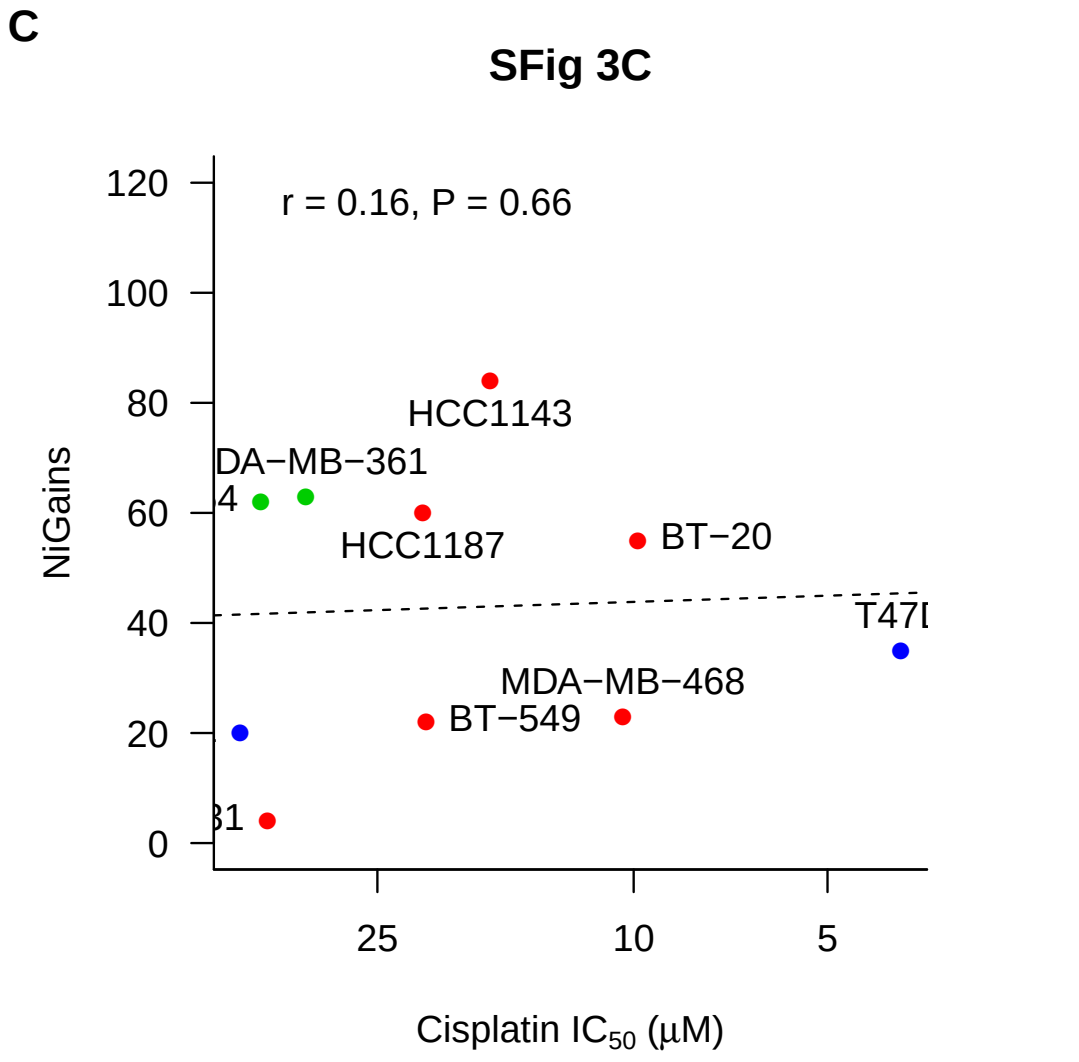
```
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("B", xoff = 2)
```



Plot supplementary figure 3C

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> CL.int.gains <- no.cna.CL[indx.cl, 5] - no.tel.cna.CL[indx.cl,
+     3]
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), CL.int.gains, pch = 16,
+     col = subtype.col, main = "SFig 3C", xlab = expression(paste("Cisplatin ",
+         IC[50], " (", mu, "M)")), ylab = "NiGains", ylim = c(0,
+         120), axes = F)
> pos.vec <- c(4, 4, 2, 1, 1, 2, 2, 3, 3, 3)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), CL.int.gains, indx.cl,
+     pos = pos.vec)
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> a <- lm(CL.int.gains ~ tmp)
> abline(a, lty = 2)
```

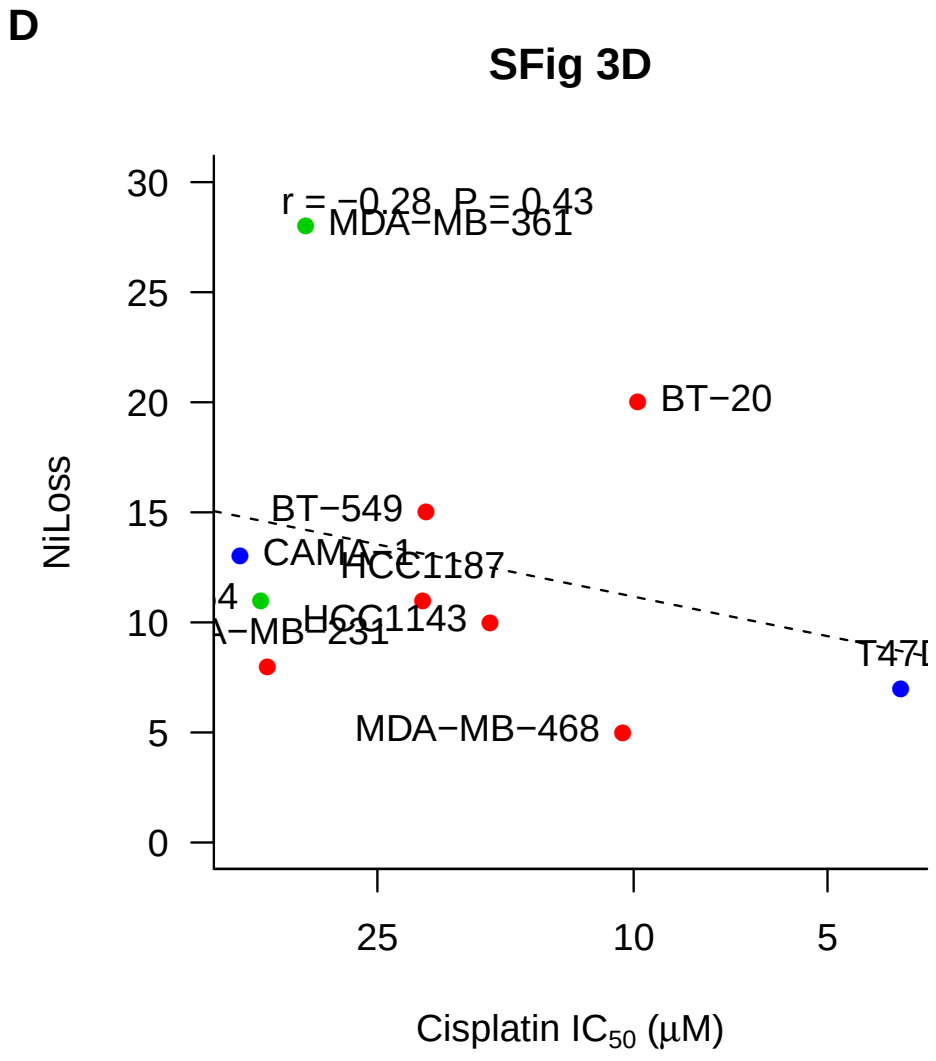
```
> a <- cor.test(CL.int.gains, tmp, method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("C", xoff = 2)
```



Plot supplementary figure 3D

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> CL.int.loss <- no.cna.CL[indx.cl, 6] - no.tel.cna.CL[indx.cl,
+     5]
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), CL.int.loss, pch = 16,
+     col = subtype.col, main = "SFig 3D", xlab = expression(paste("Cisplatin ",
+         IC[50], " (", mu, "M)")), ylab = "NiLoss", ylim = c(0,
+         30), axes = F)
```

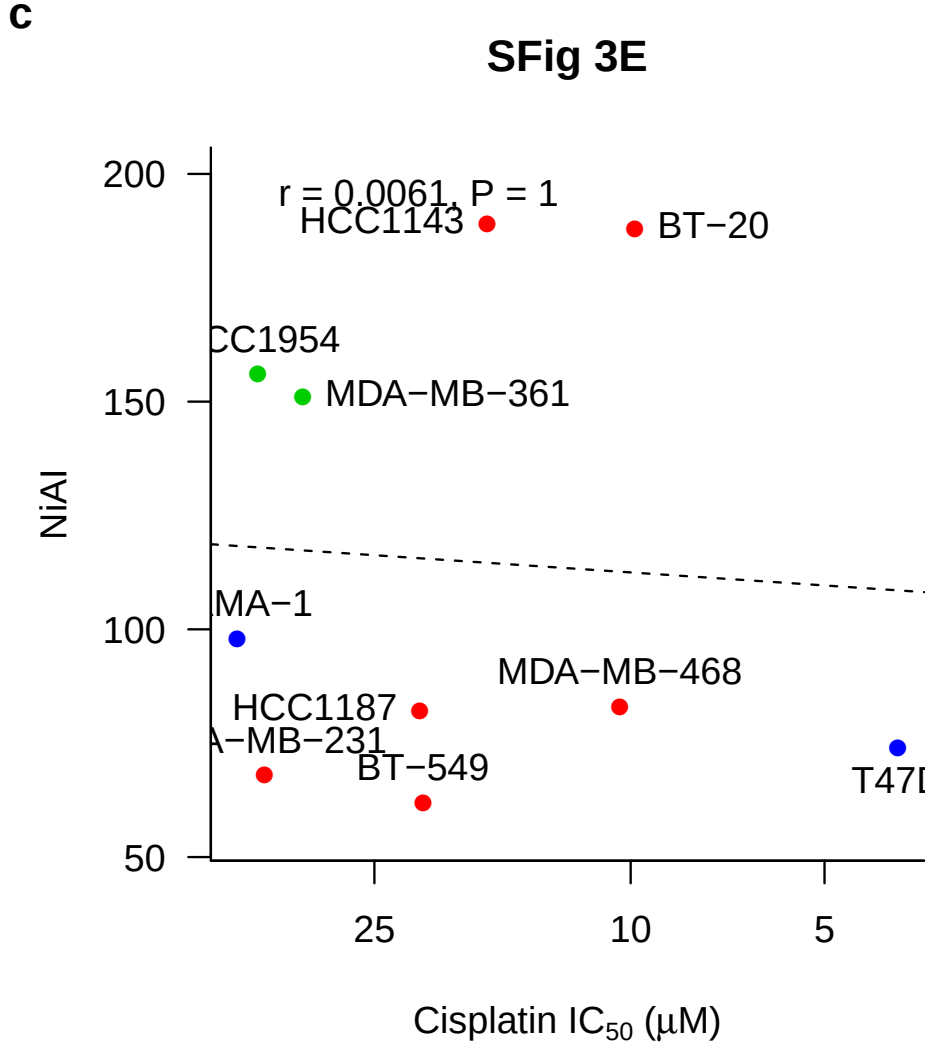
```
> pos.vec <- c(4, 2, 4, 2, 3, 2, 3, 4, 2, 3)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), CL.int.loss, indx.cl,
+     pos = pos.vec)
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> a <- lm(CL.int.loss ~ tmp)
> abline(a, lty = 2)
> a <- cor.test(CL.int.loss, tmp, method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("D", xoff = 2)
```



Plot supplementary figure 3E

```
> par(las = 1, cex = 1, bty = "l", pty = "s")
> plot(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.CL[indx.cl, 3],
+     pch = 16, col = subtype.col, main = "SFig 3E", xlab = expression(paste("Cisplatin ",
+         IC[50], " (", mu, "M)")), ylab = "NiAI", ylim = c(55,
+         200), axes = F)
> pos.vec <- c(4, 3, 3, 2, 2, 3, 3, 4, 3, 1)
> text(-log10(ic50.CL[indx.cl, 1]/1e+06), no.tel.CL[indx.cl, 3],
+     indx.cl, pos = pos.vec)
> tmp <- -log10(ic50.CL[indx.cl, 1]/1e+06)
> a <- lm(no.tel.CL[indx.cl, 3] ~ tmp)
> abline(a, lty = 2)
> a <- cor.test(no.tel.CL[indx.cl, 3], tmp, method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> box()
> axis(2)
> axis(1, at = -log10(c(1000, 100, 50, 25, 10, 5, 1)/1e+06), tick = T,
+     line = 0, labels = c(1000, 100, 50, 25, 10, 5, 1))
> label.panel("c", xoff = 2)
```

c

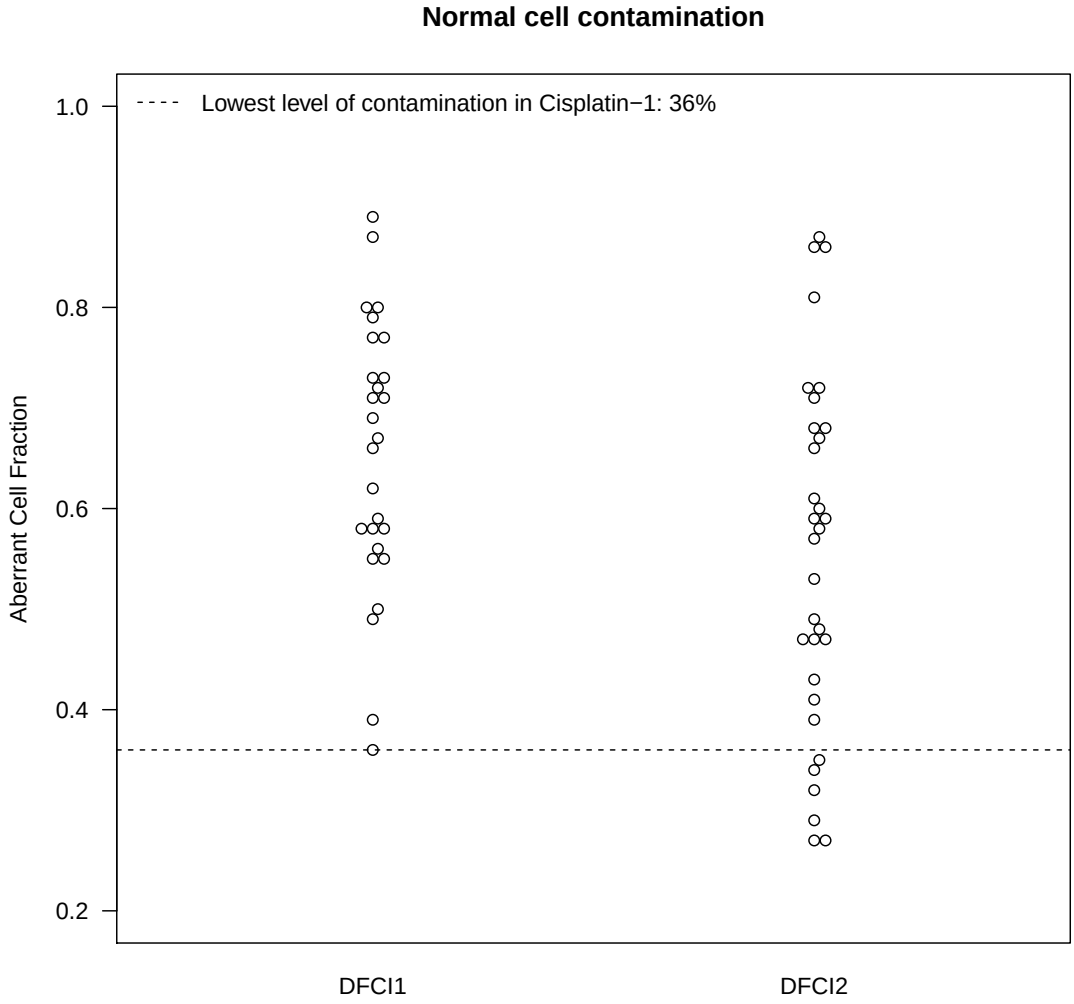

SFig 3E
r = 0.0061, P = 1
HCC1143
BT−20
CC1954
MDA−MB−361
MA−1
MDA−MB−468
HCC1187
A−MB−231
BT−549
T47I
200
150
100
50
NiAI
25
10
5
Cisplatin IC50 (μM)

4 Cisplatin clinical trials

4.1 Contamination

Compare contamination level of Cisplatin-1 and Cisplatin-2 trial SNP data. This information is returned by ASCAT

```
> trial.contam <- cbind(c(dfci1.ascat.output$aberrantcellfraction,
+     dfci2.ascat.output$aberrantcellfraction), c(rep("DFCI1",
+     length(dfci1.ascat.output$aberrantcellfraction)), rep("DFCI2",
+     length(dfci2.ascat.output$aberrantcellfraction))))
> rownames(trial.contam) <- c(colnames(dfci1.ascat.output$nA),
+     colnames(dfci2.ascat.output$nA))
> trial.contam <- trial.contam[c(colnames(dfci1.ascat.output$nA),
+     dfci2.samples), ]

> par(las = 1, cex = 1)
> beeswarm(as.numeric(trial.contam[, 1]) ~ trial.contam[, 2], xlab = "",
+     ylab = "Aberrant Cell Fraction", main = "Normal cell contamination",
+     ylim = c(0.2, 1))
> dfci1.lowest <- sort(as.numeric(trial.contam[trial.contam[, 2] ==
+     "DFCI1", 1]))[1]
> abline(h = dfci1.lowest, lty = 2)
> legend("topleft", legend = paste("Lowest level of contamination in Cisplatin-1: ",
+     dfci1.lowest * 100, "%", sep = ""), lty = 2, bty = "n", lwd = 1)
```



Make vectors of sample contamination to include

```
> dfci1.cont <- as.numeric(trial.contam[trial.contam[, 2] == "DFCI1",
+     1])
> names(dfci1.cont) <- names(trial.contam[trial.contam[, 2] ==
+     "DFCI1", 1])
> dfci1.cont

 X1T  X3T  X4T  X5T  X6T  X7T  X8T  X9T X10T X11T X12T X13T X14T X15T X16T X17T
0.80 0.71 0.49 0.58 0.69 0.87 0.73 0.71 0.56 0.58 0.72 0.77 0.67 0.36 0.62 0.39
X18T X20T X21T X22T X23T X24T X25T X26T X27T X28T X29T
0.50 0.59 0.55 0.79 0.89 0.66 0.73 0.58 0.77 0.55 0.80

> dfci2.cont <- as.numeric(trial.contam[trial.contam[, 2] == "DFCI2",
+     1])
> names(dfci2.cont) <- names(trial.contam[trial.contam[, 2] ==
+     "DFCI2", 1])
> dfci2.cont
```

```
  P3   P4   P5   P6   P7   P9  P10  P13  P14  P15  P17  P18  P20  P21  P24  P26
0.59 0.41 0.53 0.71 0.57 0.72 0.29 0.68 0.34 0.81 0.86 0.72 0.39 0.86 0.47 0.32
 P27  P28  P29  P32  P34  P35  P37  P39  P40  P41  P46  P48  P49  P51 P45L P45R
0.49 0.68 0.66 0.47 0.60 0.59 0.67 0.43 0.48 0.47 0.87 0.61 0.35 0.27 0.58 0.27
```

Make a contamination cut-off based on the lowest level of contamination found in the Cisplatin-1 trial

```
> cont.max <- sort(dfci1.cont)[1]
> cont.max

X15T
0.36
```

4.2 Figure 2, Cisplatin-1

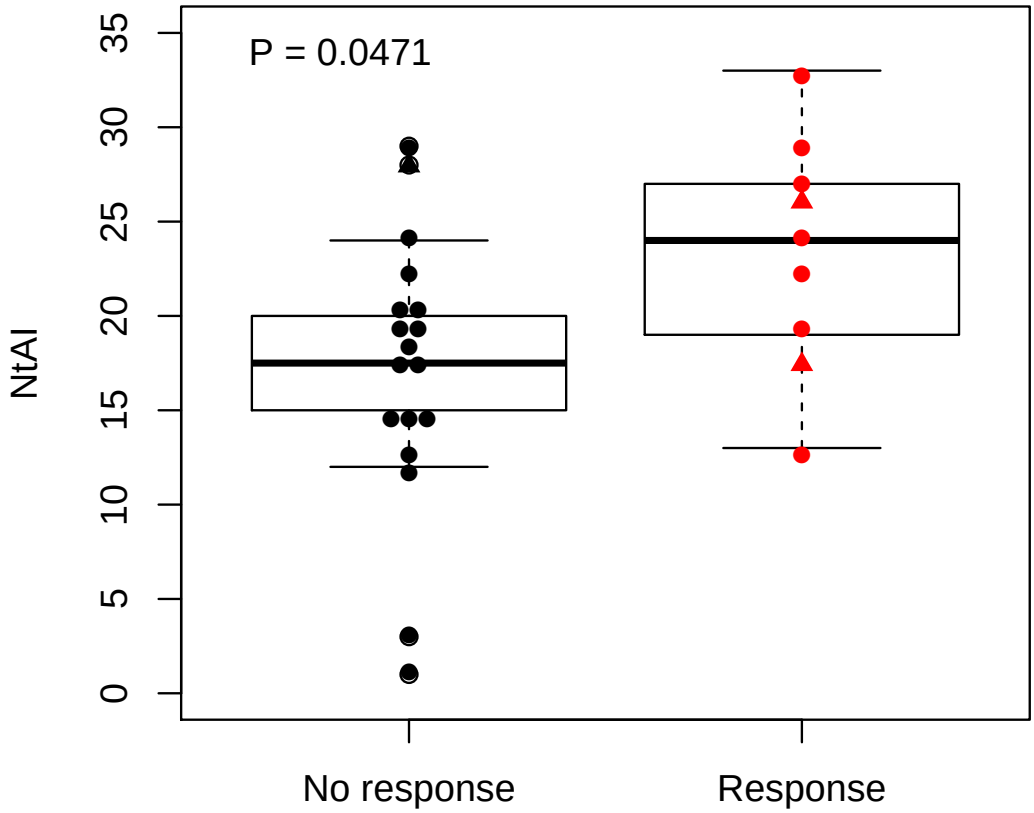
Make a binary response vector for Cisplatin-1, based on at least 90% response rates (Miller-Payne 4 and 5)

```
> dfci1.mpbin <- c(0, 1)[match(dfci1.mpgrade >= 4, c("FALSE", "TRUE"))]
> names(dfci1.mpbin) <- names(dfci1.mpgrade)
> dfci1.mpbin

 P1  P3  P4  P5  P6  P7  P8  P9 P10 P11 P12 P13 P14 P15 P16 P17 P18 P20 P21 P22
  0   1   0   1   0   1   1   1   1   0   0   0   0   0   0   1   1   0   0   0
P23 P24 P25 P26 P27 P28 P29
  0   0   0   0   0   0   1
```

Plot figure 2a

```
> boxplot(no.tel.dfci1[dfci1.cont >= cont.max, 1] ~ dfci1.mpbin[dfci1.cont >=
+     cont.max], ylab = "NtAI", xlab = "", notch = F, names = c("No response",
+     "Response"), main = "Figure 2a", ylim = c(0, 35))
> beeswarm(no.tel.dfci1[dfci1.cont >= cont.max, 1] ~ dfci1.mpbin[dfci1.cont >=
+     cont.max], col = c(1, 2), add = T, method = "center", pwpch = c(16,
+     17)[match(dfci1.brca[dfci1.cont >= cont.max], c(0, 1))])
> legend("topleft", legend = paste("P =", signif(wilcox.test(no.tel.dfci1[dfci1.cont >=
+     cont.max, 1] ~ dfci1.mpbin[dfci1.cont >= cont.max])$p.value,
+     3)), bty = "n", pch = -1)
```

Figure 2a



Calculate the ROC curve for figure 2b, and identify the optimum number of NtAI for separation between responders on non-responder.

Then determine the area under the curve and the p-value for association between sensitivity to cisplatin and high numbers of NtAI.

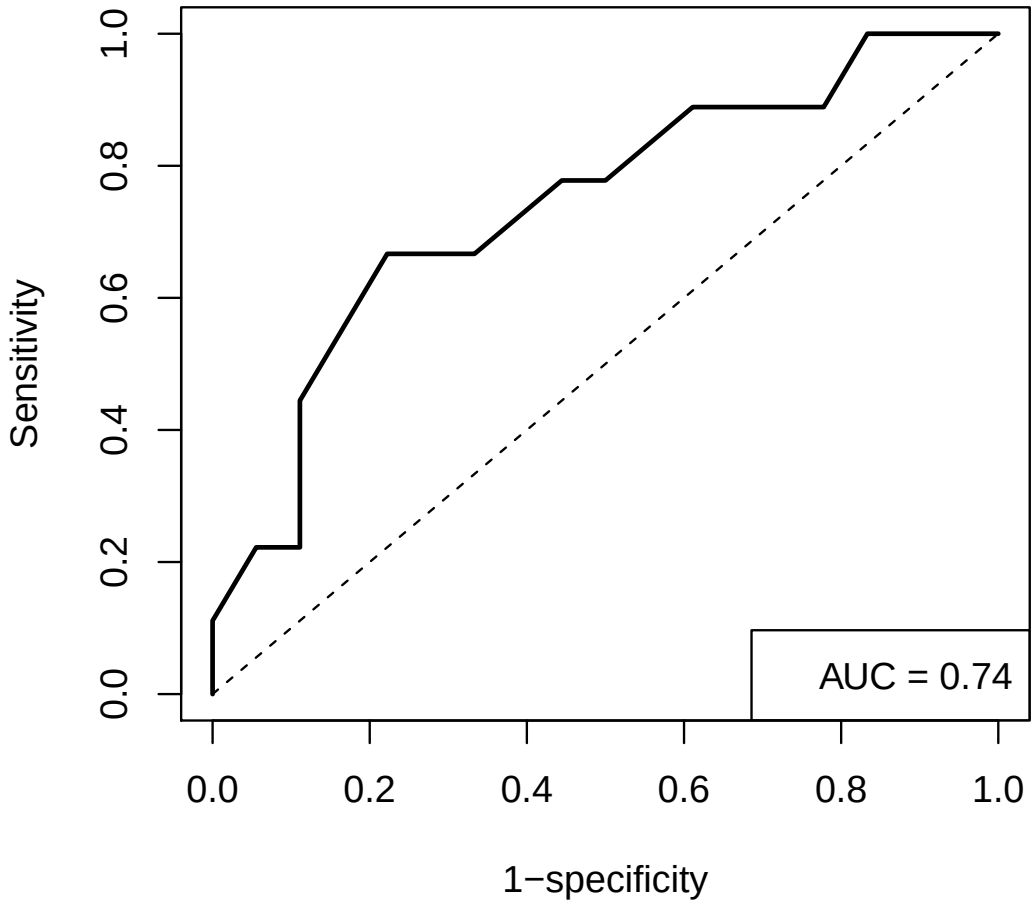
```
> R1 <- rocdemo.sca(dfci1.mpbin[dfci1.cont >= cont.max], no.tel.dfci1[dfci1.cont >=
+     cont.max, 1])
> cut.indx <- which(sqrt((1 - R1@sens)^2 + (1 - R1@spec)^2) ==
+     min(sqrt((1 - R1@sens)^2 + (1 - R1@spec)^2)))
> optimum.cut <- R1@cuts[cut.indx]
> optimum.cut

[1] 21.5

> AUC(R1)

[1] 0.7407407

> wilcox.test(no.tel.dfci1[dfci1.cont >= cont.max, 1] ~ dfci1.mpbin[dfci1.cont >=
+     cont.max])
```

```
        Wilcoxon rank sum test with continuity correction

data:  no.tel.dfci1[dfci1.cont >= cont.max, 1] by dfci1.mpbin[dfci1.cont >= cont.max]
W = 42, p-value = 0.0471
alternative hypothesis: true location shift is not equal to 0
```

Plot figure 2b

```
> plot(R1, ylab = "Sensitivity", xlab = "1-specificity", lty = 1,
+     col = 1, main = "Figure 2b", lwd = 2)
> lines(c(0, 1), c(0, 1), lty = 2)
> legend("bottomright", legend = paste("AUC = ", signif(AUC(R1),
+     2), sep = ""), pch = c(-1), col = c(-1))
```

Figure 2b



Calculate AUC confidence intervals for figure 2B

```
> ci.auc.cis1 <- ci.auc(no.tel.dfci1[dfci1.cont >= cont.max, 1],
+     dfci1.mpbin[dfci1.cont >= cont.max])
> ci.auc.cis1
```

```
      Low      High
0.5027807 0.9006955
```

4.3 Figure 2, Cisplatin-2

Tables of false/true positives/negatives for both trials based on the optimum cut-off for Cisplatin-1.

This is used to make table of ACC, PPV, NPV, SENS, SPEC values.

```
> acc.table.1 <- table(no.tel.dfci1[dfci1.cont >= cont.max, 1] >=
+     optimum.cut, dfci1.mpgrade[dfci1.cont >= cont.max] >= 4)
> acc.table.1

       FALSE TRUE
  FALSE    14    3
  TRUE      4    6

> acc.table.2 <- table(no.tel.dfci2[dfci2.cont >= cont.max, 1] >=
+     optimum.cut, dfci2.mpgrade[dfci2.cont >= cont.max] >= 4)
> acc.table.2

       FALSE TRUE
  FALSE    10    0
  TRUE      7    9

> table1 <- matrix(NA, nrow = 7, ncol = 2)
> colnames(table1) <- c("Cisplatin-1", "Cisplatin-2")
> rownames(table1) <- c("ACC", "PPV", "SENS", "NPV", "SPEC", "OR",
+     "P")
> table1[1, 1] <- signif((acc.table.1[2, 2] + acc.table.1[1, 1])/(sum(acc.table.1)),
+     2)
> table1[1, 2] <- signif((acc.table.2[2, 2] + acc.table.2[1, 1])/(sum(acc.table.2)),
+     2)
> table1[2, 1] <- signif((acc.table.1[2, 2])/(acc.table.1[2, 2] +
+     acc.table.1[2, 1]), 2)
> table1[2, 2] <- signif((acc.table.2[2, 2])/(acc.table.2[2, 2] +
+     acc.table.2[2, 1]), 2)
> table1[3, 1] <- signif((acc.table.1[2, 2])/(acc.table.1[2, 2] +
+     acc.table.1[1, 2]), 2)
> table1[3, 2] <- signif((acc.table.2[2, 2])/(acc.table.2[2, 2] +
+     acc.table.2[1, 2]), 2)
> table1[4, 1] <- signif((acc.table.1[1, 1])/(acc.table.1[1, 1] +
+     acc.table.1[1, 2]), 2)
> table1[4, 2] <- signif((acc.table.2[1, 1])/(acc.table.2[1, 1] +
+     acc.table.2[1, 2]), 2)
> table1[5, 1] <- signif((acc.table.1[1, 1])/(acc.table.1[1, 1] +
+     acc.table.1[2, 1]), 2)
> table1[5, 2] <- signif((acc.table.2[1, 1])/(acc.table.2[1, 1] +
+     acc.table.2[2, 1]), 2)
> table1[6, 1] <- paste(signif(fisher.test(acc.table.1)$estimate,
+     2), " [", signif(fisher.test(acc.table.1)$conf.int[1], 2),
+     "-", signif(fisher.test(acc.table.1)$conf.int[2], 2), "]",
+     sep = "")
> table1[6, 2] <- paste(signif(fisher.test(acc.table.2)$estimate,
+     2), " [", signif(fisher.test(acc.table.2)$conf.int[1], 2),
+     "-", signif(fisher.test(acc.table.2)$conf.int[2], 2), "]",
+     sep = "")
> table1[7, 1] <- signif(fisher.test(acc.table.1)$p.value, 2)
```

```
> table1[7, 2] <- signif(fisher.test(acc.table.2)$p.value, 2)
> table1

     Cisplatin-1    Cisplatin-2
ACC  "0.74"         "0.73"
PPV  "0.6"          "0.56"
SENS "0.67"         "1"
NPV  "0.82"         "1"
SPEC "0.78"         "0.59"
OR   "6.4 [0.9-60]" "Inf [1.9-Inf]"
P    "0.039"        "0.0039"
```

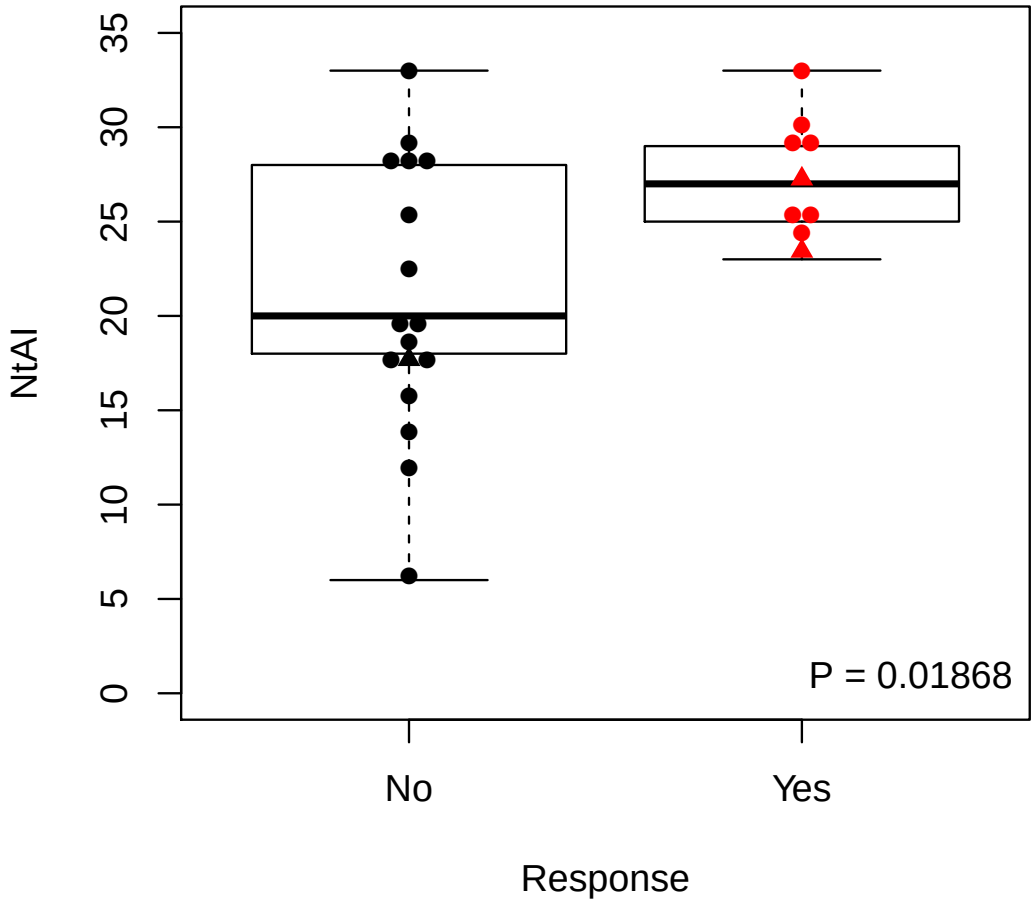
Make a binary response vector for Cisplatin-2, based on at least 90% response rates (Miller-Payne 4 and 5)

```
> dfci2.mpbin <- c(0, 1)[match(dfci2.mpgrade >= 4, c("FALSE", "TRUE"))]
> names(dfci2.mpbin) <- names(dfci2.mpgrade)
> dfci2.mpbin

  P3   P4   P5   P6   P7   P9  P10  P13  P14  P15  P17  P18  P20  P21  P24  P26
   1    0    0    1    1    0    0    0    0    1    1    0    0    0    0    1
 P27  P28  P29  P32  P34  P35  P37  P39  P40  P41  P46  P48  P49  P51 P45L P45R
   1    0    0    0    0    0    0    0    1    1    0    1    0    1    0    1
```

Plot figure 2c

```
> boxplot(no.tel.dfci2[dfci2.cont >= cont.max, 1] ~ dfci2.mpbin[dfci2.cont >=
+     cont.max], ylab = "NtAI", xlab = "Response", notch = F, names = c("No",
+     "Yes"), main = "Figure 2c", ylim = c(0, 35))
> beeswarm(no.tel.dfci2[dfci2.cont >= cont.max, 1] ~ dfci2.mpbin[dfci2.cont >=
+     cont.max], col = c(1, 2), add = T, method = "center", pwpch = c(16,
+     17)[match(dfci2.brca[dfci2.cont >= cont.max], c(0, 1))])
> legend("bottomright", legend = paste("P =", signif(wilcox.test(no.tel.dfci2[dfci2.cont >=
+     cont.max, 1] ~ dfci2.mpbin[dfci2.cont >= cont.max])$p.value,
+     4)), bty = "n", pch = -1)
```

Figure 2c



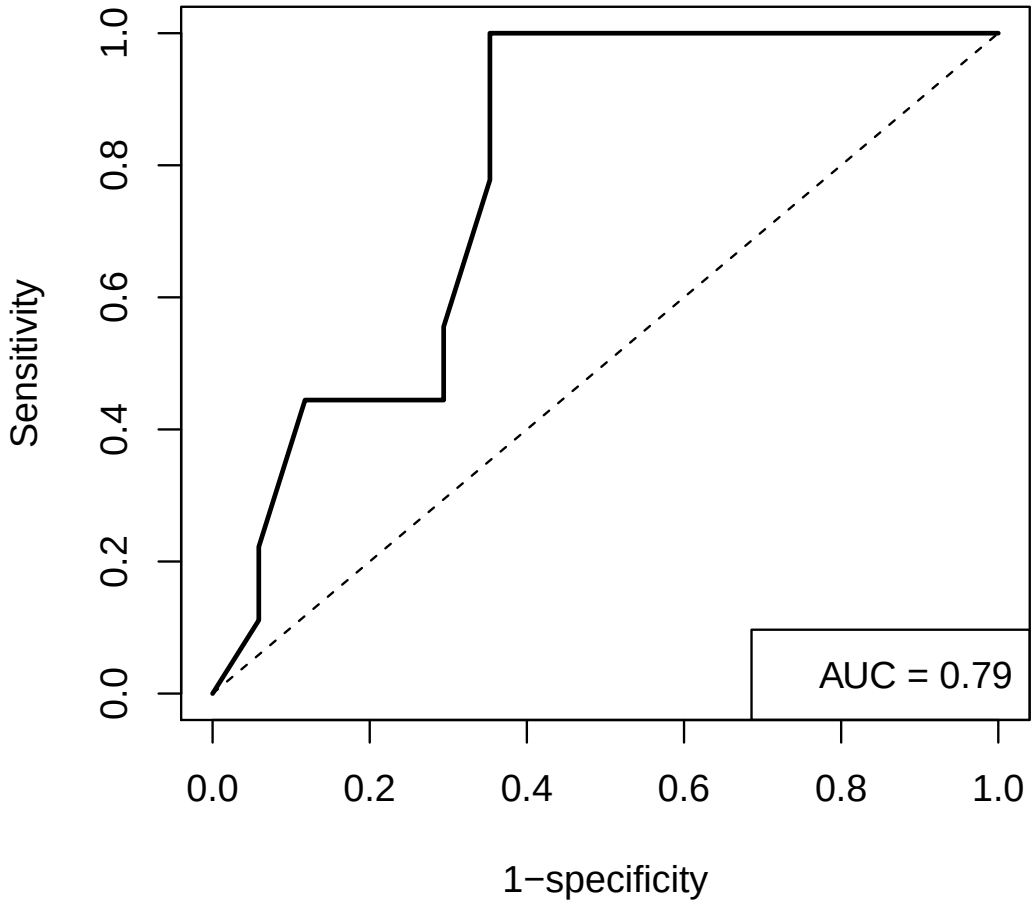
Calculate the ROC curve for figure 2f and determine the area under the curve and the p-value for association between sensitivity to cisplatin and high numbers of NtAI.

```
> R2 <- rocdemo.sca(dfci2.mpbin[dfci2.cont >= cont.max], no.tel.dfci2[dfci2.cont >=
+     cont.max, 1])
> AUC(R2)

[1] 0.7875817

> wilcox.test(no.tel.dfci2[dfci2.cont >= cont.max, 1] ~ dfci2.mpbin[dfci2.cont >=
+     cont.max])

        Wilcoxon rank sum test with continuity correction

data:  no.tel.dfci2[dfci2.cont >= cont.max, 1] by dfci2.mpbin[dfci2.cont >= cont.max]
W = 32.5, p-value = 0.01868
alternative hypothesis: true location shift is not equal to 0
```

Plot figure 2d

```
> plot(R2, ylab = "Sensitivity", xlab = "1-specificity", lty = 1,
+     col = 1, main = "Figure 2d", lwd = 2)
> lines(c(0, 1), c(0, 1), lty = 2)
> legend("bottomright", legend = c(paste("AUC = ", signif(AUC(R2),
+     2), sep = "")), pch = c(-1), col = c(-1))
```

Figure 2d



Calculate AUC confidence intervals for figure 2D

```
> ci.auc.cis2 <- ci.auc(no.tel.dfci2[dfci2.cont >= cont.max, 1],
+     dfci2.mpbin[dfci2.cont >= cont.max])
> ci.auc.cis2

      Low      High
0.5541667 0.9270833
```

5 TCGA ovarian

5.1 Figure 3

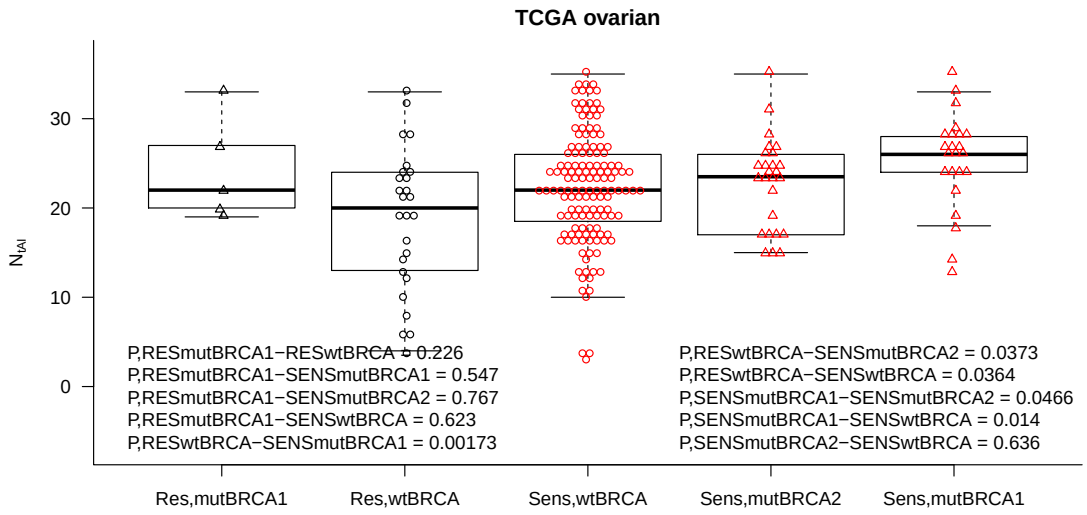
Prepare the data by organizing the samples into 5 groups: mutated BRCA1 resistant, wtBRCA1/2 resistant, wtBRCA1/2 sensitive, mutated BRCA2 sensitive, mutated BRCA1 sensitive. There are no resistant mutated BRCA2. One sample has both a BRCA1 and BRCA2 mutation, this sample is placed in both groups. These 5 groups will be named 1-5, so that we can plot them in this order

```
> tcga.ov.subset.vec <- intersect(names(tcga.ovarian.outcome[tcga.ovarian.outcome !=
+     "null" & tcga.ovarian.platinum.drug & tcga_ovarian_ascat_aberrant >=
+     cont.max]), rownames(tcga.ov.brca.mut))
> tcga.ov.wtbrca <- name.vect(tcga.ov.brca.mut[tcga.ov.subset.vec,
+     1] == "" & tcga.ov.brca.mut[tcga.ov.subset.vec, 2] %in% c("",
+     "AMP;"), rownames(tcga.ov.brca.mut[tcga.ov.subset.vec, ]))
> tcga.ov.brca1 <- name.vect(tcga.ov.brca.mut[tcga.ov.subset.vec,
+     1] == "", rownames(tcga.ov.brca.mut[tcga.ov.subset.vec, ]))
> tcga.ov.brca2 <- name.vect(tcga.ov.brca.mut[tcga.ov.subset.vec,
+     2] %in% c("", "AMP;"), rownames(tcga.ov.brca.mut[tcga.ov.subset.vec,
+     ]))
> tcga.ov.brca.sens.stat <- tcga.ov.wtbrca
> tcga.ov.brca.sens.stat[tcga.ov.outcome.180[tcga.ov.subset.vec] &
+     tcga.ov.wtbrca] <- 3
> tcga.ov.brca.sens.stat[!tcga.ov.outcome.180[tcga.ov.subset.vec] &
+     tcga.ov.wtbrca] <- 2
> tcga.ov.brca.sens.stat[tcga.ov.outcome.180[tcga.ov.subset.vec] &
+     !tcga.ov.brca1] <- 5
> tcga.ov.brca.sens.stat[!tcga.ov.outcome.180[tcga.ov.subset.vec] &
+     !tcga.ov.brca1] <- 1
> tcga.ov.brca.sens.stat[tcga.ov.outcome.180[tcga.ov.subset.vec] &
+     !tcga.ov.brca2] <- 4
> tcga.ov.brca.sens.stat[!tcga.ov.brca2 & !tcga.ov.brca1] <- 6
> tcga.ov.brca.sens.stat <- c(tcga.ov.brca.sens.stat, tcga.ov.brca.sens.stat[tcga.ov.brca.sens.st
+     6])
> tcga.ov.brca.sens.stat[tcga.ov.brca.sens.stat %in% 6] <- c(4,
+     5)
> table(tcga.ov.brca.sens.stat)

tcga.ov.brca.sens.stat
  1   2   3   4   5
  5  26 139  26  23
```

Plot figure 3

```
> par(las = 1, cex = 1, bty = "l", mar = c(4, 4, 2, 2) + 0.1)
> boxplot(no.tel.tcga[names(tcga.ov.brca.sens.stat), 1] ~ tcga.ov.brca.sens.stat,
+     ylab = expression(N[tAI]), xlab = "", notch = F, main = "TCGA ovarian",
+     pch = 16, ylim = c(-7, 37), outline = F, names = c("Res,mutBRCA1",
+         "Res,wtBRCA", "Sens,wtBRCA", "Sens,mutBRCA2", "Sens,mutBRCA1"))
> beeswarm(no.tel.tcga[names(tcga.ov.brca.sens.stat), 1] ~ tcga.ov.brca.sens.stat,
+     col = c(1, 1, 2, 2, 2), add = T, pch = c(2, 1, 1, 2, 2),
+     method = "hex", cex = 0.8)
> legend("bottomleft", legend = c(paste("P,RESmutBRCA1-RESwtBRCA =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
```

```
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(1, 2))$p.value, 3)), paste("P,RESmutBRCA1-SENSmutBRCA1 =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(1, 5))$p.value, 3)), paste("P,RESmutBRCA1-SENSmutBRCA2 =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(1, 4))$p.value, 3)), paste("P,RESmutBRCA1-SENSwtBRCA =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(1, 3))$p.value, 3)), paste("P,RESwtBRCA-SENSmutBRCA1 =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(2, 5))$p.value, 3))), bty = "n", pch = c(-1), col = c(-1))
> legend("bottomright", legend = c(paste("P,RESwtBRCA-SENSmutBRCA2 =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(2, 4))$p.value, 3)), paste("P,RESwtBRCA-SENSwtBRCA =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(2, 3))$p.value, 3)), paste("P,SENSmutBRCA1-SENSmutBRCA2 =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(5, 4))$p.value, 3)), paste("P,SENSmutBRCA1-SENSwtBRCA =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(5, 3))$p.value, 3)), paste("P,SENSmutBRCA2-SENSwtBRCA =",
+     signif(wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat),
+         1] ~ tcga.ov.brca.sens.stat, subset = tcga.ov.brca.sens.stat %in%
+         c(4, 3))$p.value, 3))), bty = "n", pch = c(-1), col = c(-1))
```



5.2 Supplementary figure 4

Calculate the ROC curve for supplementary figure 4.

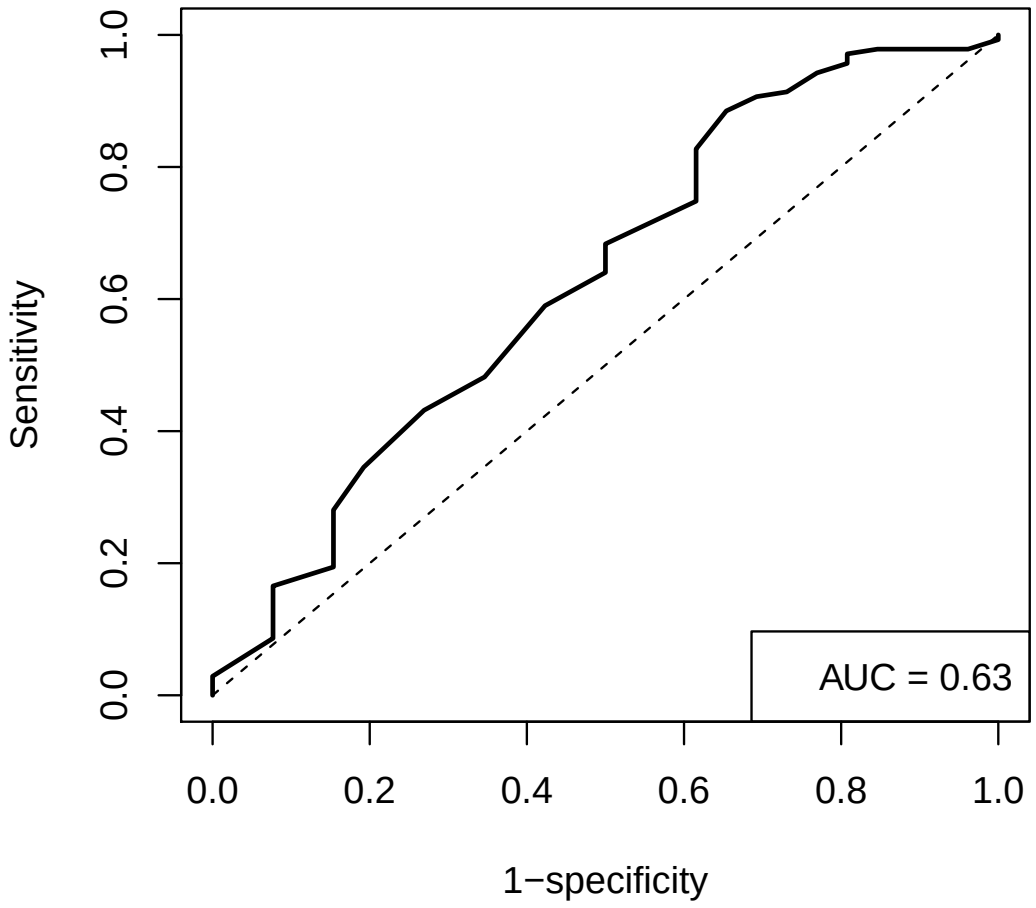
Then determine the area under the curve and the p-value for association between sensitivity to cisplatin and high numbers of NtAI in the wtBRCA samples.

```
> R3 <- rocdemo.sca(c(0, 1)[match(tcga.ov.outcome.180[names(tcga.ov.brca.sens.stat[tcga.ov.brca.s
+     c(2, 3)])], c("FALSE", "TRUE"))], no.tel.tcga[names(tcga.ov.brca.sens.stat[tcga.ov.brca.sen
+     c(2, 3)]), 1])
> AUC(R3)

[1] 0.6293581

> wilcox.test(no.tel.tcga[names(tcga.ov.brca.sens.stat[tcga.ov.brca.sens.stat %in%
+     c(2, 3)]), 1] ~ tcga.ov.outcome.180[names(tcga.ov.brca.sens.stat[tcga.ov.brca.sens.stat %in
+     c(2, 3)])])

        Wilcoxon rank sum test with continuity correction

data:  no.tel.tcga[names(tcga.ov.brca.sens.stat[tcga.ov.brca.sens.stat %in% c(2, 3)]), 1] by tcga
W = 1339.5, p-value = 0.03644
alternative hypothesis: true location shift is not equal to 0
```

Plot supplementary figure 4

```
> plot(R3, ylab = "Sensitivity", xlab = "1-specificity", lty = 1,
+     col = 1, main = "SFig 4", lwd = 2)
> lines(c(0, 1), c(0, 1), lty = 2)
> legend("bottomright", legend = paste("AUC = ", signif(AUC(R3),
+     2), sep = ""), pch = c(-1), col = c(-1))
```

SFig 4



Calculate AUC confidence intervals for supplementary figure 4

```
> ci.auc.tcga <- ci.auc(no.tel.tcga[names(tcga.ov.brca.sens.stat[tcga.ov.brca.sens.stat %in%
+     c(2, 3)]), 1], c(0, 1)[match(tcga.ov.outcome.180[names(tcga.ov.brca.sens.stat[tcga.ov.brca.
+     c(2, 3)])], c("FALSE", "TRUE"))])
> ci.auc.tcga

      Low      High
0.5004431 0.7600515
```

6 NtAI breakpoints

6.1 NtAI breakpoint location

Organize data

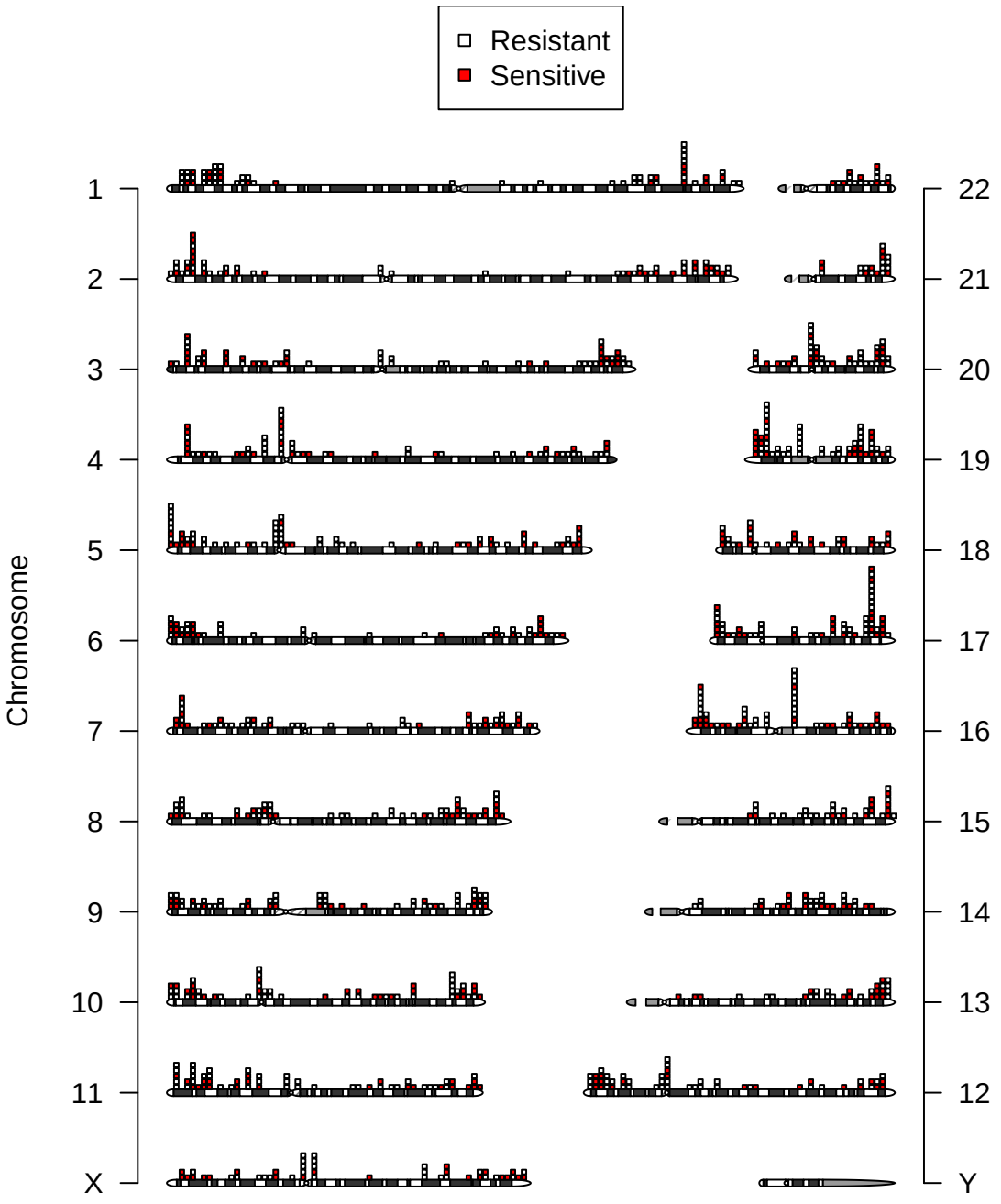
```
> dfci1.telAI.seg <- telAI.fn(dfci1.ascat.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cis1)
> dfci2.telAI.seg <- telAI.fn(dfci2.ascat.seg, chrominfo = chrominfo,
+     min.probes = min.probes.cis2)
> dfci1.names <- dfci1.mpgrade[dfci1.cont >= cont.max]
> names(dfci1.names) <- names(dfci1.cont[dfci1.cont >= cont.max])
> dfci2.names <- dfci2.mpgrade[dfci2.cont >= cont.max]
> names(dfci2.names) <- names(dfci2.cont[dfci2.cont >= cont.max])
> comb.tel.seg <- rbind(dfci1.telAI.seg, dfci2.telAI.seg)
> comb.tel.seg <- comb.tel.seg[comb.tel.seg[, 1] %in% names(c(dfci1.names,
+     dfci2.names)), ]
> telAI.breaks <- matrix(0, 0, 4)
> for (i in 1:nrow(comb.tel.seg)) {
+     a <- c(2, 4)[match(comb.tel.seg[i, 1] %in% names(dfci1.names),
+         c("TRUE", "FALSE"))]
+     b <- c(0, 2)[match(comb.tel.seg[i, 1] %in% names(c(dfci1.names,
+         dfci2.names)[c(dfci1.names, dfci2.names) %in% c(4:5)]),
+         c("FALSE", "TRUE"))]
+     telAI.breaks <- rbind(telAI.breaks, c(as.numeric(comb.tel.seg[i,
+         2]), NA, a, b))
+     if (as.numeric(comb.tel.seg[i, 3]) < chrominfo[as.numeric(comb.tel.seg[i,
+         2]), 3] * 1000) {
+         telAI.breaks[i, 2] <- as.numeric(comb.tel.seg[i, 4])
+     }
+     if (as.numeric(comb.tel.seg[i, 3]) > chrominfo[as.numeric(comb.tel.seg[i,
+         2]), 3] * 1000) {
+         telAI.breaks[i, 2] <- as.numeric(comb.tel.seg[i, 3])
+     }
+ }
> CHR <- telAI.breaks[, 1]
> CHR[CHR == 23] <- "X"
> MapInfo <- telAI.breaks[, 2]
> rownames(telAI.breaks) <- 1:nrow(telAI.breaks)
```

6.2 Supplementary figure 5

Plot supplementary figure 5

```
> par(las = 1)
> chrompos <- prepareGenomePlot(data.frame(CHR, MapInfo), paintCytobands = TRUE,
+     organism = "hsa", sexChromosomes = T)
> a <- beeswarm(chrompos[, 2] ~ chrompos[, 1], horizontal = T,
+     vertical = F, do.plot = F, method = "square", pch = 22, pwbg = telAI.breaks[,
+         4], add = T, cex = 0.4)
> a <- apply(a, 2, as.numeric)
> rownames(a) <- 1:nrow(a)
> bb <- sort(as.numeric(names(table(as.numeric(a[, 1]) - as.numeric(a[,
+     6])))))
> bb <- bb[bb > 0][1]
> a[, 1] <- a[, 1] + bb
> for (i in unique(a[, 6])) {
+     b <- a[a[, 6] == i, ]
+     if (nrow(b[b[, 1] <= b[, 6], , drop = F]) > 0) {
+         tmp <- b[as.numeric(b[, 1]) <= as.numeric(b[, 6]), ,
+             drop = F]
+         tmp <- tmp[order(abs(as.numeric(tmp[, 1]) - as.numeric(tmp[,
+             6])), decreasing = T), , drop = F]
+         tmp <- tmp[!duplicated(tmp[, 2]), , drop = F]
+         for (k in 1:nrow(tmp)) {
+             ab <- as.numeric(tmp[k, 6]) - as.numeric(tmp[k, 1])
+             b[b[, 2] %in% tmp[k, 2], 1] <- b[b[, 2] %in% tmp[k,
+                 2], 1] + ab + bb
+         }
+         a[rownames(b), 1] <- b[, 1]
+     }
+ }
> points(a[, 2], a[, 1], cex = 0.4, pch = 22, bg = a[, 5])
> label.panel("A", xoff = 2)
> legend("top", legend = c("Resistant", "Sensitive"), col = 1,
+     pch = 22, pt.bg = c(0, 2))
```

A


Resistant
Sensitive
Chromosome
1
2
3
4
5
6
7
8
9
10
11
X
22
21
20
19
18
17
16
15
14
13
12
Y

6.3 NtAI breakpoints associated with CNVs

Prepare to make figures 4a-b

Load location data on common CNVs.

Data on common CNVs has been acquired from the Database of Genomic Variants.

Cisplatin-1 mapping is based on build 17 of the human genome, while cisplatin-2 is based on build 18.

Load and organize data

```
> tmp <- read.table("DGV/variation.hg18.v10.nov.2010.txt", sep = "\t",
+     as.is = T, header = T)
> tmp <- tmp[tmp[, 6] == "CopyNumber", ]
> tmp <- tmp[, c(3:5)]
> tmp[, 1] <- sub("chr", "", tmp[, 1])
> tmp[tmp[, 1] %in% "X", 1] <- 23
> tmp <- tmp[tmp[, 1] %in% c(1:23), ]
> commonCNV.18 <- apply(tmp, 2, as.numeric)
> tmp <- read.table("DGV/variation.hg17.v10.nov.2010.txt", sep = "\t",
+     as.is = T, header = T)
> tmp <- tmp[tmp[, 6] == "CopyNumber", ]
> tmp <- tmp[, c(3:5)]
> tmp[, 1] <- sub("chr", "", tmp[, 1])
> tmp[tmp[, 1] %in% "X", 1] <- 23
> tmp <- tmp[tmp[, 1] %in% c(1:23), ]
> commonCNV.17 <- apply(tmp, 2, as.numeric)
```

Determine the number of NtAIs associated with CNVs within 25 kb at either side, in all samples

```
> window.size <- 25000
> CNV.cis1.breaks <- telBreak.CNV.fn(dfci1.telAI.seg[dfci1.telAI.seg[,
+     1] %in% names(dfci1.names), ], commonCNV.17, window.size = window.size)
> CNV.cis1.breaks

Associated with           Total
            255             517

> CNV.cis2.breaks <- telBreak.CNV.fn(dfci2.telAI.seg[dfci2.telAI.seg[,
+     1] %in% names(dfci2.names), ], commonCNV.18, window.size = window.size)
> CNV.cis2.breaks

Associated with           Total
            340             599
```

To test if the breakpoints are significantly associated with CNVs, permutations must be performed where the location of the breakpoints are randomly shuffled, and then tested for association with CNVs

Probe location is based on the SNP chips, and can be obtained from the GEO, either from the data we uploaded (GSE28330), or from the OncoScan platform entry. When picking random breakpoints, it is important to restrict the analysis to actual SNP probe locations, as these are the only possible breakpoints in the real data.

Load probe locations

```
> cis2mark <- read.table("MIPCNReference/mip_cn_330K_markers.txt",
+     header = T, as.is = T, row.names = 1, sep = "\t")
> cis2mark[, 4] <- sub("chr", "", cis2mark[, 4])
```

```
> cis2mark[cis2mark[, 4] == "X", 4] <- 23
> cis2mark <- cis2mark[!cis2mark[, 4] == "Y", ]
> cis2mark <- cis2mark[!cis2mark[, 2] == cis2mark[, 3], ]
> cis2mark <- apply(cis2mark[, 4:5], 2, as.numeric)
> cis1mark <- read.table("MIPCNReference/mip1.markers.txt", header = T,
+     as.is = T, row.names = 1, sep = "\t")
> cis1mark[cis1mark[, 3] == "X", 3] <- 23
> cis1mark <- apply(cis1mark[, c(3:4)], 2, as.numeric)
> cis1mark <- cis1mark[order(cis1mark[, 2]), ]
> cis1mark <- cis1mark[order(cis1mark[, 1]), ]
> head(cis1mark)

     Chrom. Chrom..Position
[1,]      1         2201241
[2,]      1         2779554
[3,]      1         2905810
[4,]      1         3097040
[5,]      1         3117988
[6,]      1         3128433

> head(cis2mark)

     Chrom Chrom.Pos..bp.
[1,]     1          524446
[2,]     1          744055
[3,]     1          788822
[4,]     1         1038818
[5,]     1         1148140
[6,]     1         1211299
```

A function to perform the permutations

```
> perm.fn <- function(data.type, window.size, probe.location, no.breaks = 1000,
+     no.perm = 1000) {
+     output <- list()
+     for (j in 1:no.perm) {
+         cat(j)
+         tmp1 <- probe.location[sample(1:nrow(probe.location),
+             no.breaks, replace = T), ]
+         tmp2 <- vector()
+         for (i in 1:no.breaks) {
+             tmp <- data.type[data.type[, 1] == tmp1[i, 1], ,
+                 drop = F]
+             a <- c(tmp1[i, 2] - window.size, tmp1[i, 2] + window.size)
+             tmp <- tmp[tmp[, 3] >= a[1], , drop = F]
+             tmp <- tmp[tmp[, 2] <= a[2], , drop = F]
+             tmp2[i] <- nrow(tmp)
+         }
+         output[[j]] <- tmp2
+     }
+     return(output)
+ }
```

Now do permutations to test for significance.

This step takes a very long time, and has therefore been performed separately, and is now loaded as an R object.

For clarity, the code used to perform permutations is shown below.

perm.cnv.cis1.all <- perm.fn(commonCNV.17, window.size=window.size, probe.location=cis1mark, no.perm=1000, no.breaks=CNV.cis1.breaks[2])

perm.cnv.cis2.all <- perm.fn(commonCNV.18, window.size=window.size, probe.location=cis2mark, no.perm=1000, no.breaks=CNV.cis2.breaks[2])

perm.list.36cont.20111020 <- list(perm.cnv.cis1.all, perm.cnv.cis2.all)

names(perm.list.36cont.20111020) <- c("perm.cnv.cis1.all", "perm.cnv.cis2.all")

save(perm.list.36cont.20111020, file="perm.list.36cont.20111020.RData")

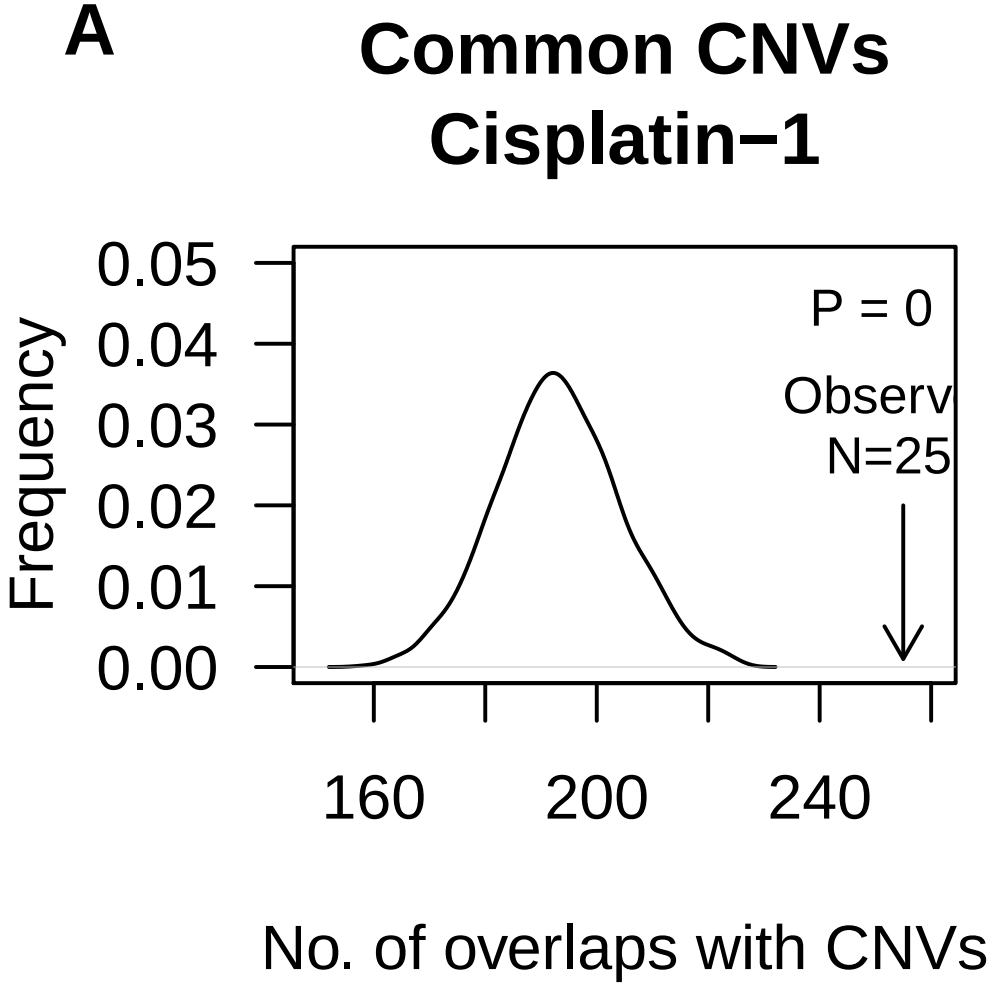
Load object containing permutated data

```
> load("perm.list.36cont.20111020.RData")
```

6.4 Figure 4

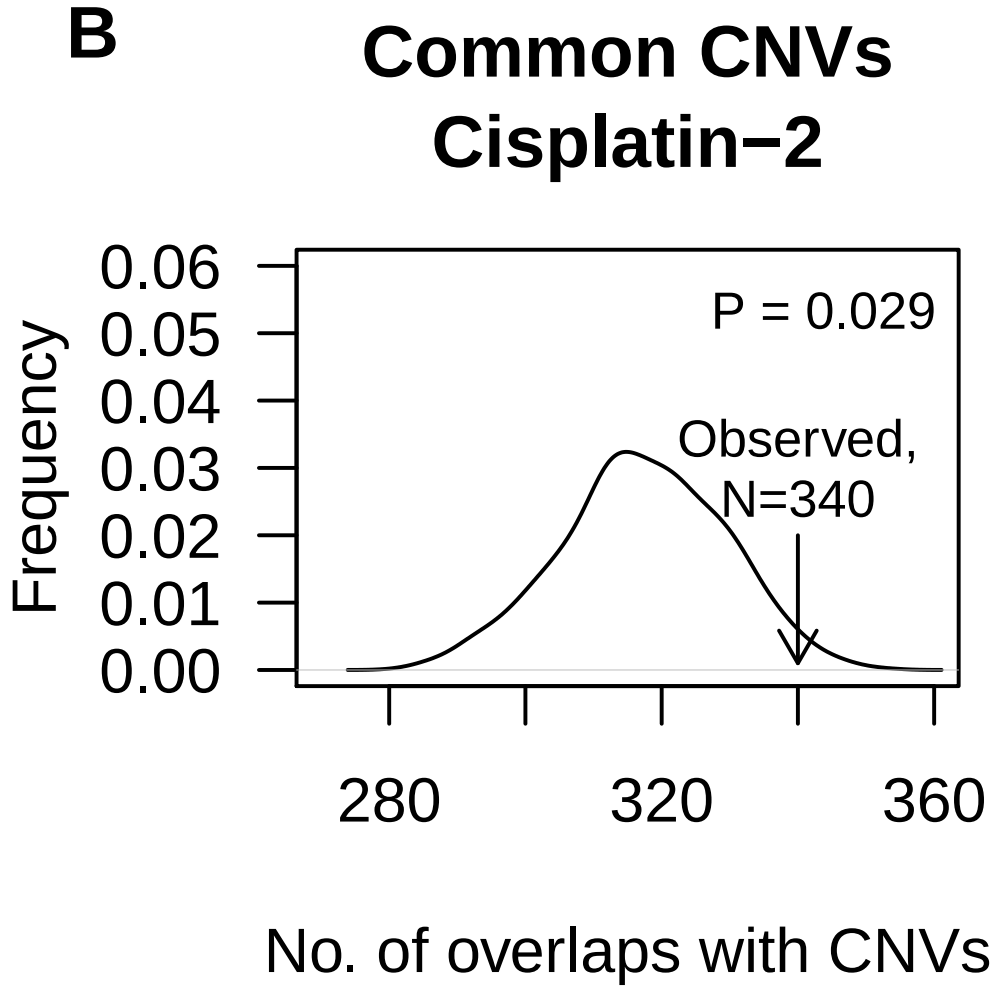
Plot figure 4a

```
> par(las = 1)
> tmp <- sapply(perm.list.36cont.20111020$perm.cnv.cis1.all, function(x) {
+     length(x[x != 0])
+ })
> P1 <- length(tmp[tmp >= CNV.cis1.breaks[1]])/1001
> plot(density(tmp), xlim = c(150, 260), ylim = c(0, 0.05), xlab = "No. of overlaps with CNVs",
+     ylab = "Frequency", main = "Common CNVs\nCisplatin-1")
> arrows(CNV.cis1.breaks[1], 0.02, CNV.cis1.breaks[1], 0.001, lwd = 1,
+     col = 1, length = 0.1)
> text(CNV.cis1.breaks[1], 0.03, paste("Observed,\nN=", CNV.cis1.breaks[1],
+     sep = ""), cex = 0.8)
> legend("topright", legend = paste("P =", signif(P1, 2)), bty = "n",
+     lty = -1, lwd = -1, cex = 0.8)
> label.panel("A", xoff = 2)
```



Plot figure 4B

```
> par(las = 1)
> tmp <- sapply(perm.list.36cont.20111020$perm.cnv.cis2.all, function(x) {
+     length(x[x != 0])
+ })
> P1 <- length(tmp[tmp >= CNV.cis2.breaks[1]])/1001
> plot(density(tmp), xlim = c(270, 360), ylim = c(0, 0.06), xlab = "No. of overlaps with CNVs",
+     ylab = "Frequency", main = "Common CNVs\nCisplatin-2")
> arrows(CNV.cis2.breaks[1], 0.02, CNV.cis2.breaks[1], 0.001, lwd = 1,
+     col = 1, length = 0.1)
> text(CNV.cis2.breaks[1], 0.03, paste("Observed,\nN=", CNV.cis2.breaks[1],
+     sep = ""), cex = 0.8)
> legend("topright", legend = paste("P =", signif(P1, 2)), bty = "n",
+     lty = -1, lwd = -1, cex = 0.8)
> label.panel("B", xoff = 2)
```



7 BRCA1 expression versus NtAI

7.1 BRCA1 in Cisplat-1 and Cisplatin-2, by qPCR

We have BRCA1 methylation and qPCR expression on only a subset of the trial samples. These do not entirely overlap with the samples for which we have SNP data, as the RNA/DNA content was insufficient in some cases. Therefore these data will be loaded and handled separately, so that we can use as many samples as possible. Ie, when we plot figure 5B (BRCA1 expression versus BRCA1 methylation) and SFig 6A (optimizing specificity and sensitivity via ROC curves), some samples will be plotted for which we do not have SNP data.

```
> brca1.methylation <- read.table("BRCA1_Methyltation.txt", sep = "\t",
+     header = T, as.is = T)
> brca1.methylation
```

```
         Study Sample Methylation Mutation
1  Cisplatin 1     P1           1
2  Cisplatin 1     P3           0
3  Cisplatin 1     P4           0
4  Cisplatin 1     P6           0
5  Cisplatin 1     P7           1
6  Cisplatin 1     P8           1
7  Cisplatin 1    P10           0
8  Cisplatin 1    P11           0
9  Cisplatin 1    P12           0
10 Cisplatin 1    P14           0
11 Cisplatin 1    P15           1
12 Cisplatin 1    P16           0
13 Cisplatin 1    P17           1
14 Cisplatin 1    P20           0
15 Cisplatin 1    P21           0
16 Cisplatin 1    P22           0
17 Cisplatin 1    P23           1
18 Cisplatin 1    P24           0    BRCA2
19 Cisplatin 1    P25           1
20 Cisplatin 1    P26           1
21 Cisplatin 1    P27           0
22 Cisplatin 1    P28           0
23 Cisplatin 1    P29           0
24 Cisplatin 2     P3           0
25 Cisplatin 2     P4           1
26 Cisplatin 2     P6           1
27 Cisplatin 2     P9           0
28 Cisplatin 2    P10           0
29 Cisplatin 2    P13           1
30 Cisplatin 2    P14           0
31 Cisplatin 2    P15           0    BRCA1
32 Cisplatin 2    P17           0
33 Cisplatin 2    P20           1    BRCA1
34 Cisplatin 2    P30           1
35 Cisplatin 2    P32           0
36 Cisplatin 2    P37           0
37 Cisplatin 2    P48           1
38 Cisplatin 2    P49           0
```

```
> brca1.qpcr <- read.table("brca1_qpcr.txt", sep = "\t", header = T,
+     as.is = T)
> brca1.qpcr
```

	Study	Sample	MP.score	BRCA	Expression	Average.CT.B1.e16.e17	Average.CT.36B4
1	Cisplatin 1	P3	5		0.028164077	32.4	27.25
2	Cisplatin 1	P6	1		0.030536249	32.2	27.17
3	Cisplatin 1	P7	4		0.000405878	35.2	23.97
4	Cisplatin 1	P8	4		0.008668512	32.8	25.95
5	Cisplatin 1	P10	4		0.009843133	33.8	
6	Cisplatin 1	P12	1		0.014578640	30.7	
7	Cisplatin 1	P2	4		0.025382887	32.8	
8	Cisplatin 1	P11	3		0.003817031	34.8	
9	Cisplatin 1	P15	0		0.027840585	32.3	
10	Cisplatin 1	P4	1		0.014245664	33.6	
11	Cisplatin 1	P5	5	BRCA1	0.020380501	33.0	
12	Cisplatin 1	P14	2		0.035896824	33.2	
13	Cisplatin 1	P16	1		0.056328154	32.5	
14	Cisplatin 1	P23	3		0.001586430	34.9	
15	Cisplatin 1	P28	2		0.019236631	32.4	
16	Cisplatin 1	P22	2		0.000573998	36.2	
17	Cisplatin 1	P25	3		0.031613108	33.4	
18	Cisplatin 1	P21	0		0.018367936	32.7	
19	Cisplatin 1	P24	2	BRCA2	0.070153878	30.8	
20	Cisplatin 1	P29	5		0.002192309	34.4	
21	Cisplatin 1	P17	5		0.005274804	32.4	
22	Cisplatin 1	P20	2		0.012545670	33.9	
23	Cisplatin 2	P14	1		0.000051700	28.7	
24	Cisplatin 2	P10	1		0.000090900	28.6	
25	Cisplatin 2	P49	1		0.000030200	29.1	
26	Cisplatin 2	P19	2		0.000085200	28.3	
27	Cisplatin 2	P12	1		0.000010600	31.0	
28	Cisplatin 2	P30	1		0.000009910	31.1	
29	Cisplatin 2	P2	1		0.000043900	29.0	
30	Cisplatin 2	P11	3		0.000252795	26.8	
31	Cisplatin 2	P23	2		0.000041100	29.5	
32	Cisplatin 2	P32	1		0.000198318	27.2	
33	Cisplatin 2	P3	4		0.000028800	29.7	
34	Cisplatin 2	P4	3		0.000030400	30.2	
35	Cisplatin 2	P37	3		0.000116658	27.4	
36	Cisplatin 2	P15	5	BRCA1	0.000086700	28.4	
37	Cisplatin 2	P41	5		0.000008350	31.9	
38	Cisplatin 2	P9	1		0.000033900	29.4	
39	Cisplatin 2	P20	2	BRCA1	0.000105089	28.7	
40	Cisplatin 2	P13	3		0.000030500	29.3	
41	Cisplatin 2	P27	4		0.000068600	28.3	
42	Cisplatin 2	P48	4		0.000011400	31.0	
43	Cisplatin 2	P17	5		0.000002280	32.9	
44	Cisplatin 2	P6	4		0.000003390	32.8	

```
5           27.17
6           24.60
7           27.50
8           26.77
9           27.13
10          27.47
11          27.33
12          28.40
13          28.30
14          25.63
15          26.67
16          25.47
17          28.37
18          26.93
19          26.97
20          25.53
21          24.80
22          27.53
23          14.48
24          15.17
25          14.11
26          14.78
27          14.51
28          14.45
29          14.56
30          14.83
31          14.98
32          14.85
33          14.64
34          15.16
35          14.37
36          14.90
37          15.00
38          14.57
39          15.51
40          14.30
41          14.50
42          14.60
43          14.21
44          14.66
```

The BRCA1 qPCR expression data is normalized to the housekeeping gene RPLP0. Average raw cycle numbers for each gene is in columns 6 and 7. Note that the BRCA1 expression levels for Cisplatin 1 and Cisplatin 2 are not directly comparable in their raw form, as the RNA from Cisplatin 1 was significantly more diluted, and run two years earlier on a different PCR machine.

As the data points are few in each trial individually, we combine the two trials by centering the expression data, and dividing by the standard deviation. Assuming the distribution of the expression values are similar in each trial, this will allow us to combine them.

Make vectors that hold the expression of each trial, and make sure the naming convention matches the NtAI data

```
> dfci1.brca1.meth <- brca1.methylation[brca1.methylation$Study ==
+     "Cisplatin 1", "Methylation"]
> names(dfci1.brca1.meth) <- paste(sub("P", "X", brca1.methylation[brca1.methylation$Study ==
```

```
+     "Cisplatin 1", "Sample"]), "T", sep = "")
> dfci2.brca1.meth <- brca1.methylation[brca1.methylation$Study ==
+     "Cisplatin 2", "Methylation"]
> names(dfci2.brca1.meth) <- brca1.methylation[brca1.methylation$Study ==
+     "Cisplatin 2", "Sample"]
> dfci1.brca1.qpcr <- brca1.qpcr[brca1.qpcr$Study == "Cisplatin 1",
+     "Expression"]
> names(dfci1.brca1.qpcr) <- paste(sub("P", "X", brca1.qpcr[brca1.qpcr$Study ==
+     "Cisplatin 1", "Sample"]), "T", sep = "")
> dfci2.brca1.qpcr <- brca1.qpcr[brca1.qpcr$Study == "Cisplatin 2",
+     "Expression"]
> dfci2.brca1.mpscore.qpcr <- brca1.qpcr[brca1.qpcr$Study == "Cisplatin 2",
+     "MP.score"]
> dfci2.brca1.brcamut.qpcr <- brca1.qpcr[brca1.qpcr$Study == "Cisplatin 2",
+     "BRCA"]
> names(dfci2.brca1.mpscore.qpcr) <- names(dfci2.brca1.brcamut.qpcr) <- names(dfci2.brca1.qpcr) <
+     "Cisplatin 2", "Sample"]
```

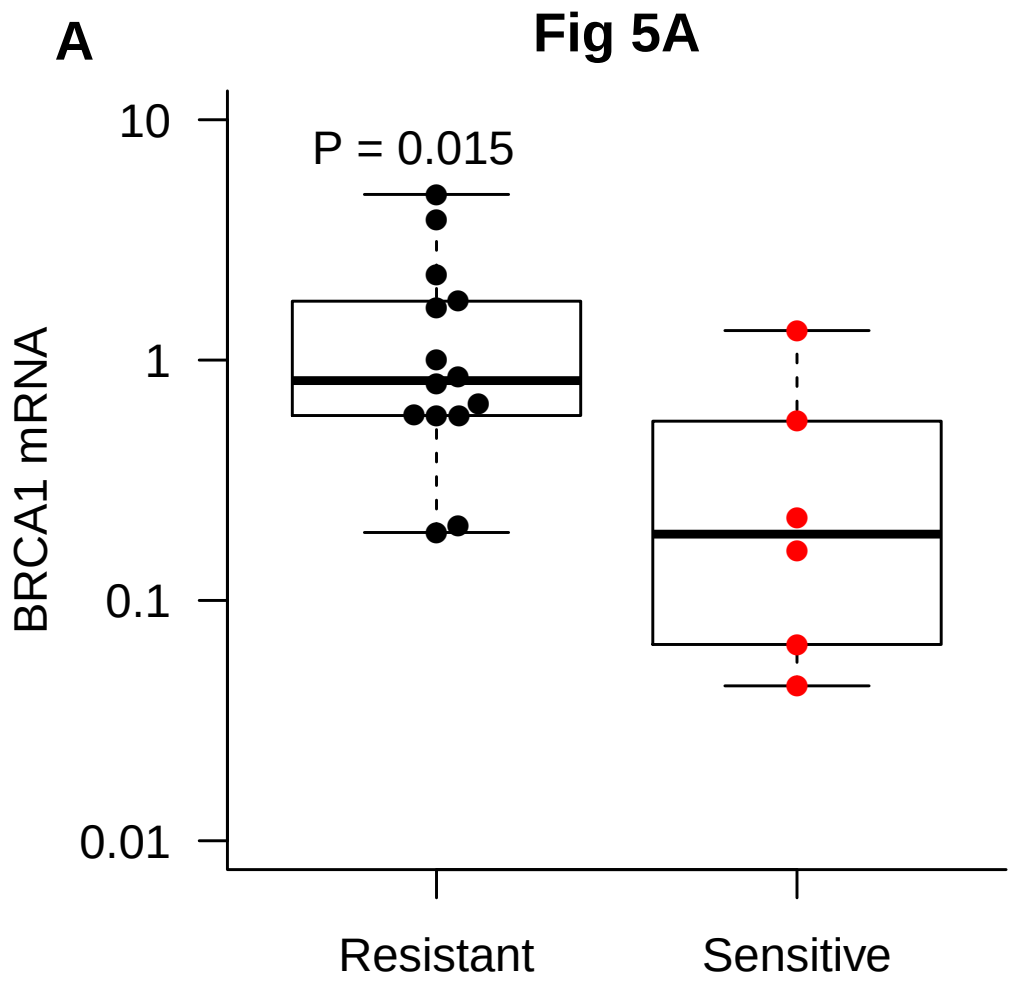
7.2 Figure 5

Now prepare to plot figure 5.

```
> dfci.comb.ntai <- c(no.tel.dfci1[dfci1.cont >= cont.max, 1],
+     no.tel.dfci2[dfci2.cont >= cont.max, 1])
> dfci.comb.qpcr <- c(scale(log2(dfci1.brca1.qpcr))[, 1], scale(log2(dfci2.brca1.qpcr))[,
+     1])
> dfci.comb.brca1meth <- c(dfci1.brca1.meth, dfci2.brca1.meth)
> dfci.comb.brcamut <- c(dfci1.brca[dfci1.cont >= cont.max], dfci2.brca[dfci2.cont >=
+     cont.max])
> names(dfci.comb.brcamut) <- c(paste(sub("P", "X", names(dfci1.brca[dfci1.cont >=
+     cont.max])), "T", sep = ""), names(dfci2.brca[dfci2.cont >=
+     cont.max]))
> dfci.comb.mp <- c(dfci1.mpgrade[dfci1.cont >= cont.max], dfci2.mpgrade[dfci2.cont >=
+     cont.max])
> names(dfci.comb.mp) <- c(paste(sub("P", "X", names(dfci1.mpgrade[dfci1.cont >=
+     cont.max])), "T", sep = ""), names(dfci2.mpgrade[dfci2.cont >=
+     cont.max]))
> dfci.comb.mp.qpcr <- brca1.qpcr[, "MP.score"]
> names(dfci.comb.mp.qpcr) <- c(names(dfci1.brca1.qpcr), names(dfci2.brca1.qpcr))
> dfci.comb.brcamut.qpcr <- c(0, 1)[match(brca1.qpcr[, "BRCA"] ==
+     "", c("TRUE", "FALSE"))]
> names(dfci.comb.brcamut.qpcr) <- c(names(dfci1.brca1.qpcr), names(dfci2.brca1.qpcr))
> dfci.comb.brca1meth.qpcr <- c("Negative", "Positive", "Unknown")[match(dfci.comb.brca1meth[name
+     c(0, 1, NA))]
> names(dfci.comb.brca1meth.qpcr) <- c(names(dfci1.brca1.qpcr),
+     names(dfci2.brca1.qpcr))
```

Now we can plot figure 5A, BRCA1 expression by qPCR in Cisplatin-2, versus response. No BRCA1 mutation carriers

```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> indx <- dfci2.brca1.brcamut.qpcr == ""
> boxplot(log10(dfci2.brca1.qpcr[indx]/dfci2.brca1.qpcr[indx][1]) ~
+     dfci2.brca1.mpscore.qpcr[indx] > 3, main = "Fig 5A", ylab = "BRCA1 mRNA",
+     xlab = "", names = c("Resistant", "Sensitive"), axes = F,
+     ylim = c(-2, 1))
> beeswarm(log10(dfci2.brca1.qpcr[indx]/dfci2.brca1.qpcr[indx][1]) ~
+     dfci2.brca1.mpscore.qpcr[indx] > 3, pch = 16, col = c(1,
+     2), add = T)
> a <- signif(wilcox.test(log10(dfci2.brca1.qpcr[indx]/dfci2.brca1.qpcr[indx][1]) ~
+     dfci2.brca1.mpscore.qpcr[indx] > 3)$p.value, 2)
> legend("topleft", legend = paste("P = ", a, sep = ""), bty = "n",
+     pch = -1)
> axis(1, at = c(1, 2), labels = c("Resistant", "Sensitive"))
> box()
> axis(2, at = log10(c(0.01, 0.1, 1, 10)), labels = c(0.01, 0.1,
+     1, 10))
> label.panel("A", xoff = 2)
```



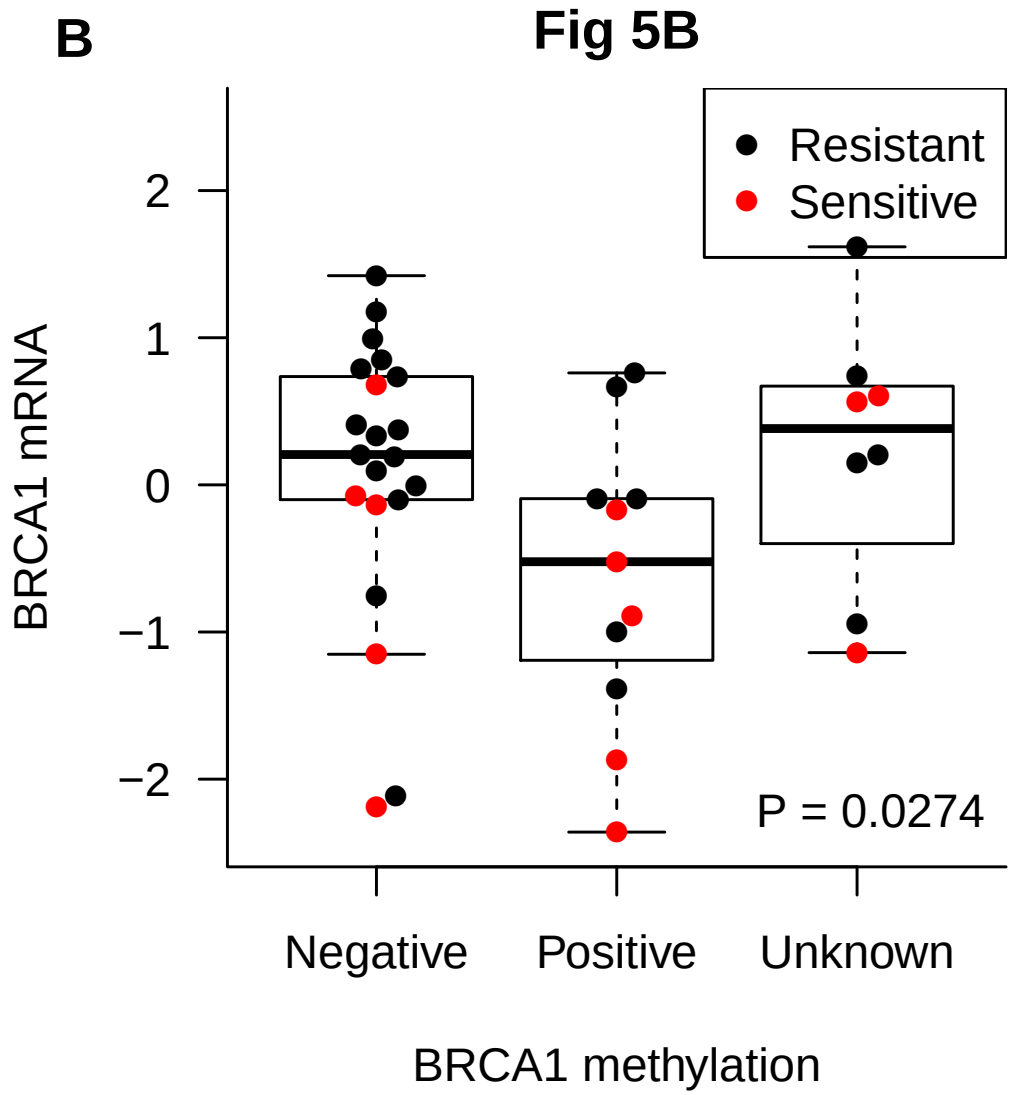
And here comes figure 5B, BRCA1 expression by qPCR in Cisplatin-2, versus BRCA1 methylation. No BRCA1 mutation carriers

Figure 5B

Now plot panel B, BRCA1 expression by qPCR versus methylation

```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> boxplot(dfci.comb.qpcr[dfci.comb.brcamut.qpcr == 0] ~ dfci.comb.brca1meth.qpcr[dfci.comb.brcamu
+     0], ylim = c(-2.4, 2.5), ylab = "BRCA1 mRNA", xlab = "BRCA1 methylation",
+     main = "Fig 5B", outline = F)
> beeswarm(dfci.comb.qpcr[dfci.comb.brcamut.qpcr == 0] ~ dfci.comb.brca1meth.qpcr[dfci.comb.brcam
+     0], add = T, pwcol = c(1, 2)[match(dfci.comb.mp.qpcr[dfci.comb.brcamut.qpcr ==
+     0] > 3, c("FALSE", "TRUE"))], pch = c(16, 16, 16))
> legend("bottomright", legend = paste("P =", signif(wilcox.test(dfci.comb.qpcr[dfci.comb.brcamut
+     0] ~ dfci.comb.brca1meth.qpcr[dfci.comb.brcamut.qpcr == 0],
+     subset = dfci.comb.brca1meth.qpcr[dfci.comb.brcamut.qpcr ==
+         0] != "Unknown")$p.value, 3)), bty = "n")
> legend("topright", legend = c("Resistant", "Sensitive"), col = c(1,
+     2), pch = c(16, 16))
```

```
> label.panel("B", xoff = 2)
```

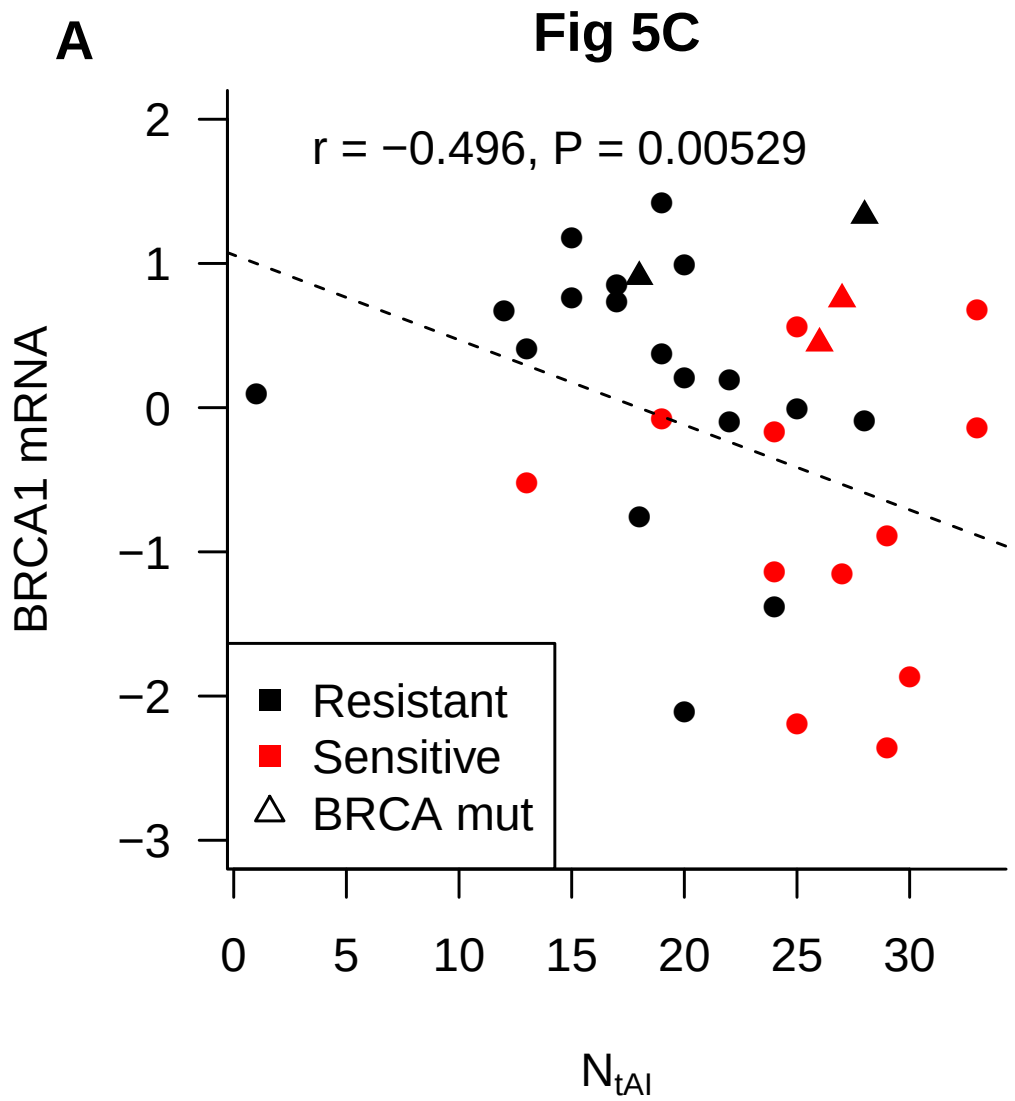


And finally we can plot figure 5C, NtAI versus BRCA1 qPCR, with colors representing response, and shapes representing BRCA1/2 mutation carriers

Figure 5C

```
> indx <- intersect(names(dfci.comb.ntai), names(dfci.comb.qpcr))
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> plot(dfci.comb.ntai[indx], dfci.comb.qpcr[indx], col = c(1, 2)[match(dfci.comb.mp[indx] >
+     3, c("FALSE", "TRUE"))], pch = c(16, 17)[match(dfci.comb.brcamut[indx],
+     c(0, 1))], main = "Fig 5C", ylab = "BRCA1 mRNA", xlab = expression(N[tAI]),
+     ylim = c(-3, 2))
> legend("bottomleft", legend = c("Resistant", "Sensitive", "BRCA mut"),
+     col = c(1, 2, 1), pch = c(15, 15, 2))
> a <- cor.test(dfci.comb.ntai[indx][dfci.comb.brcamut[indx] ==
+     0], dfci.comb.qpcr[indx][dfci.comb.brcamut[indx] == 0], method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 3),
+     ", P = ", signif(a$p.value, 3), sep = ""), bty = "n", pch = -1)
```

```
> abline(lm(dfci.comb.qpcr[indx][dfci.comb.brcamut[indx] == 0] ~
+     dfci.comb.ntai[indx][dfci.comb.brcamut[indx] == 0]), lty = 2)
> label.panel("A", xoff = 2)
```



And quickly test if methylation is significantly associated with response:

```
> meth.indx <- intersect(intersect(names(dfci.comb.brca1meth),
+     names(dfci.comb.mp.qpcr)), names(dfci.comb.brcamut.qpcr[dfci.comb.brcamut.qpcr ==
+     0]))
> fisher.test(table(dfci.comb.mp.qpcr[meth.indx] > 3, dfci.comb.brca1meth[meth.indx]))

        Fisher's Exact Test for Count Data

data:  table(dfci.comb.mp.qpcr[meth.indx] > 3, dfci.comb.brca1meth[meth.indx])
p-value = 0.2515
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.4260148 16.3891269
sample estimates:
```

```
odds ratio
  2.579584
```

It is not.

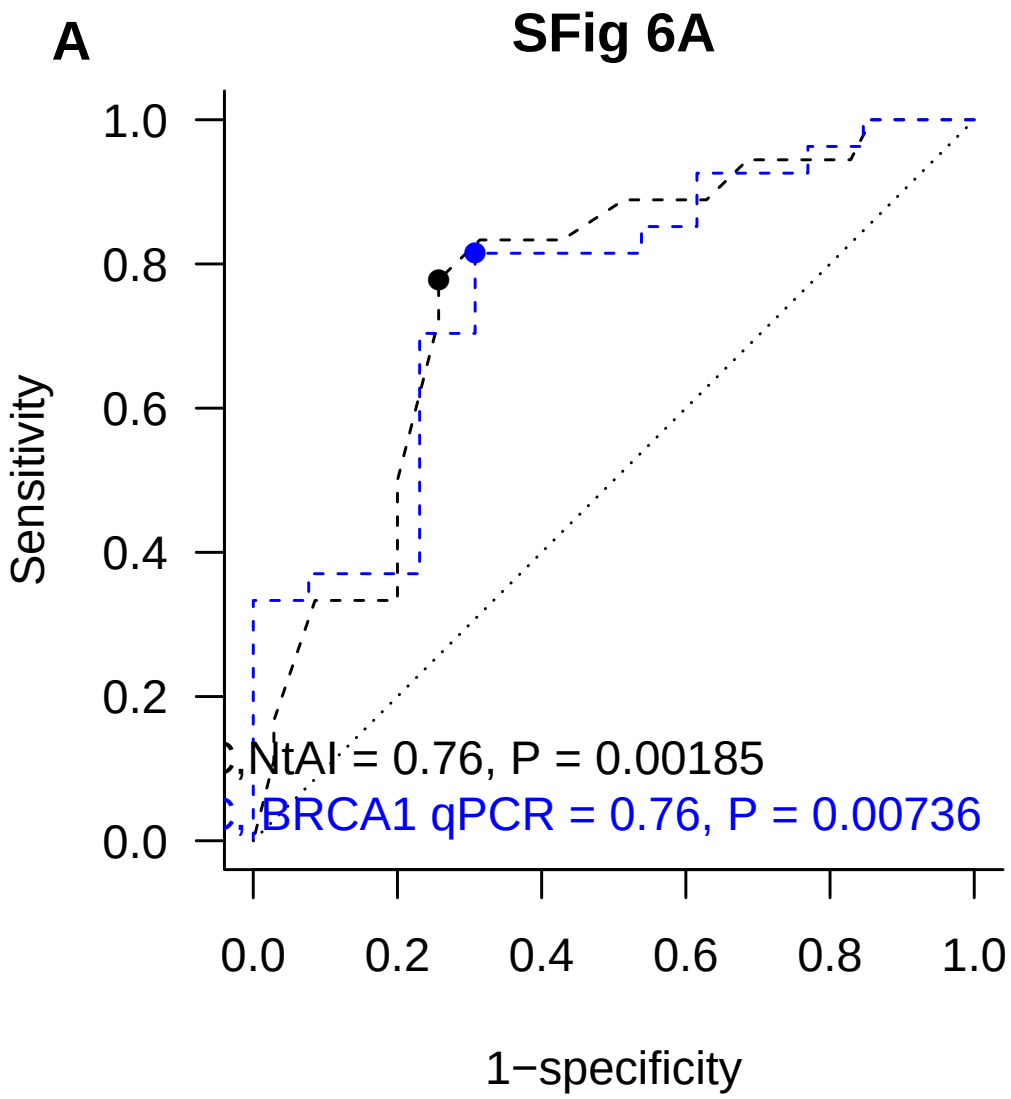
7.3 Supplementary figure 6

Now prepare for supplementary figure 6. Here we use ROC curves to find the optimum threshold of both NtAI and BRCA1 expression, that gives the best prediction of cisplatin response.

```
> R.NtAI <- rocdemo.sca(c(0, 1)[match(dfci.comb.mp > 3, c("FALSE",
+     "TRUE"))], dfci.comb.ntai)
> P.NtAI <- signif(wilcox.test(dfci.comb.ntai ~ c(0, 1)[match(dfci.comb.mp >
+     3, c("FALSE", "TRUE"))])$p.value, 3)
> R.qpcr <- rocdemo.sca(c(1, 0)[match(dfci.comb.mp.qpcr[dfci.comb.brcamut.qpcr ==
+     0] > 3, c("FALSE", "TRUE"))], dfci.comb.qpcr[dfci.comb.brcamut.qpcr ==
+     0])
> P.qpcr <- signif(wilcox.test(dfci.comb.qpcr[dfci.comb.brcamut.qpcr ==
+     0] ~ c(1, 0)[match(dfci.comb.mp.qpcr[dfci.comb.brcamut.qpcr ==
+     0] > 3, c("FALSE", "TRUE"))])$p.value, 3)
> cut.indx.NtAI <- which(sqrt((1 - R.NtAI@sens)^2 + (1 - R.NtAI@spec)^2) ==
+     min(sqrt((1 - R.NtAI@sens)^2 + (1 - R.NtAI@spec)^2)))
> optimum.cut.NtAI <- R.NtAI@cuts[cut.indx.NtAI]
> cut.indx.qpcr <- which(sqrt((1 - R.qpcr@sens)^2 + (1 - R.qpcr@spec)^2) ==
+     min(sqrt((1 - R.qpcr@sens)^2 + (1 - R.qpcr@spec)^2)))
> optimum.cut.qpcr <- R.qpcr@cuts[cut.indx.qpcr]
```

Now plot supplementary figure 6A

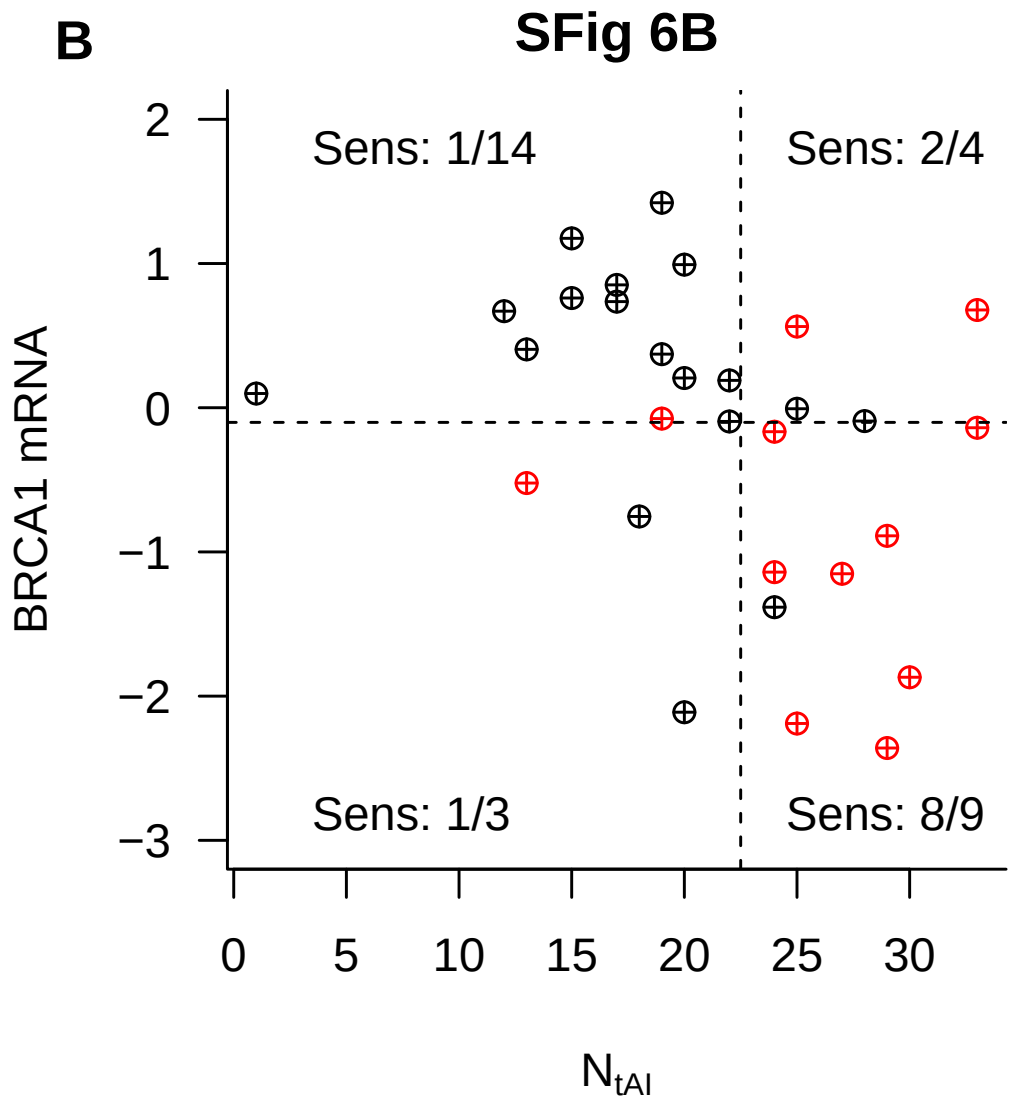
```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> plot(R.NtAI, ylab = "Sensitivity", xlab = "1-specificity", lty = 2,
+     col = 1, lwd = 1, main = "SFig 6A")
> points(R.qpcr, type = "l", lty = 2, col = 4)
> lines(c(0, 1), c(0, 1), lty = 3)
> legend("bottomright", legend = c(paste("AUC,NtAI = ", signif(AUC(R.NtAI),
+     2), ", P = ", P.NtAI, sep = ""), paste("AUC, BRCA1 qPCR = ",
+     signif(AUC(R.qpcr), 2), ", P = ", P.qpcr, sep = "")), lty = c(2,
+     2), col = c(1, 4), lwd = c(1, 1), text.col = c(1, 4), bty = "n")
> points((1 - R.NtAI@spec[cut.indx.NtAI]), R.NtAI@sens[cut.indx.NtAI],
+     pch = 16, col = 1)
> points((1 - R.qpcr@spec[cut.indx.qpcr]), R.qpcr@sens[cut.indx.qpcr],
+     pch = 16, col = 4)
> label.panel("A", xoff = 2)
```



Now plot supplementary figure 6B, only with overlap between samples with NtAI and BRCA1 qpcr expression. No BRCA1/2 mutation carriers.

```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> indx <- intersect(names(dfci.comb.ntai), names(dfci.comb.brcamut.qpcr[dfci.comb.brcamut.qpcr ==
+     0]))
> plot(dfci.comb.ntai[indx], dfci.comb.qpcr[indx], pch = 10, col = c(1,
+     2)[match(dfci.comb.mp[indx] > 3, c("FALSE", "TRUE"))], ylab = "BRCA1 mRNA",
+     xlab = expression(N[tAI]), main = "SFig 6B", ylim = c(-3,
+         2))
> abline(h = optimum.cut.qpcr, v = optimum.cut.NtAI, lty = 2)
> a <- dfci.comb.mp[indx] > 3
> llq <- table(dfci.comb.ntai[indx] < optimum.cut.NtAI & dfci.comb.qpcr[indx] <
+     optimum.cut.qpcr, a)
> lrq <- table(dfci.comb.ntai[indx] > optimum.cut.NtAI & dfci.comb.qpcr[indx] <
+     optimum.cut.qpcr, a)
> ulq <- table(dfci.comb.ntai[indx] < optimum.cut.NtAI & dfci.comb.qpcr[indx] >
+     optimum.cut.qpcr, a)
```

```
> urq <- table(dfci.comb.ntai[indx] > optimum.cut.NtAI & dfci.comb.qpcr[indx] >
+     optimum.cut.qpcr, a)
> legend("topleft", legend = paste("Sens: ", ulq[2, 2], "/", sum(ulq[2,
+     ]), sep = ""), bty = "n", pch = -1)
> legend("topright", legend = paste("Sens: ", urq[2, 2], "/", sum(urq[2,
+     ]), sep = ""), bty = "n", pch = -1)
> legend("bottomleft", legend = paste("Sens: ", llq[2, 2], "/",
+     sum(llq[2, ]), sep = ""), bty = "n", pch = -1)
> legend("bottomright", legend = paste("Sens: ", lrq[2, 2], "/",
+     sum(lrq[2, ]), sep = ""), bty = "n", pch = -1)
> label.panel("B", xoff = 2)
```



Now make a table showing ACC, PPV, NPV, SENS, RES, and p-valus for predicting cisplatin response using: NtAI alone, BRCA1 expression alone, and then combined. Use only the overlapping samples, so it is comparable.

```
> table1 <- cbind(acc.calc(dfci.comb.ntai[indx] > optimum.cut.NtAI,
+     dfci.comb.mp[indx] > 3), acc.calc(dfci.comb.qpcr[indx] <
+     optimum.cut.qpcr, dfci.comb.mp[indx] > 3), acc.calc(dfci.comb.ntai[indx] >
```

```
+     optimum.cut.NtAI & dfci.comb.qpcr[indx] < optimum.cut.qpcr,
+     dfci.comb.mp[indx] > 3))
> colnames(table1) <- c("NtAI", "BRCA1qPCR", "NtAI+BRCA1qPCR")
> table1

             NtAI   BRCA1qPCR NtAI+BRCA1qPCR
ACC  0.8333333333 0.800000000   0.8333333333
PPV  0.7692307692 0.750000000   0.8888888889
NPV  0.8823529412 0.833333333   0.8095238095
SENS 0.8333333333 0.750000000   0.6666666667
SPEC 0.8333333333 0.833333333   0.9444444444
P    0.0005367241 0.002409426   0.0006381425

> write.table(table1, file = "table.sfig6.txt", sep = "\t", quote = F)
```

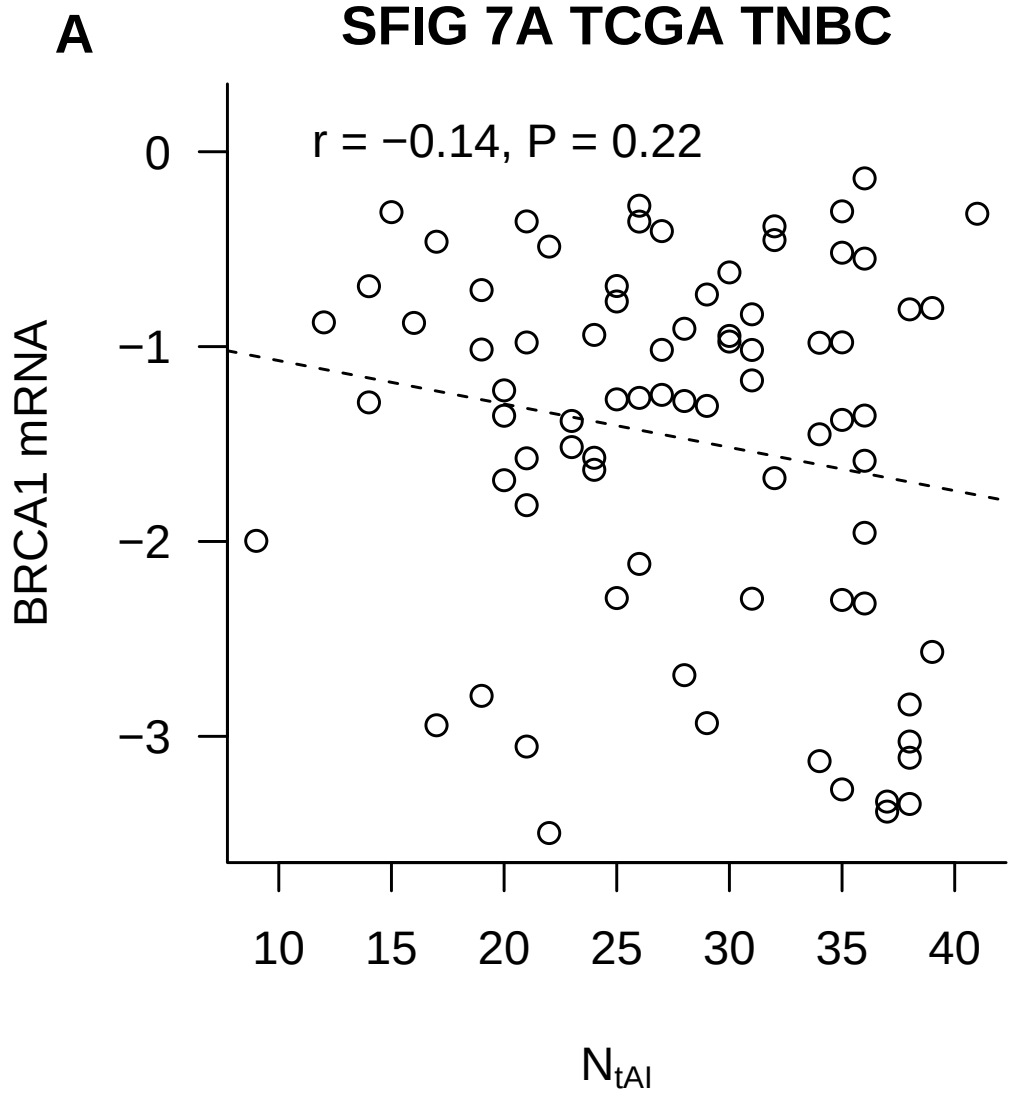
7.4 BRCA1 expression in TCGA TNBC and Ovarian cohorts

Here we will plot the correlation between BRCA1 mRNA by micro array, and NtAI, in the TCGA TNBC and ovarian cohorts. We know BRCA1/2 mutation status for the ovarian cohort, thus this is based only on the wtBRCA samples.

First in the TNBC cohort

7.5 Supplementary figure 7

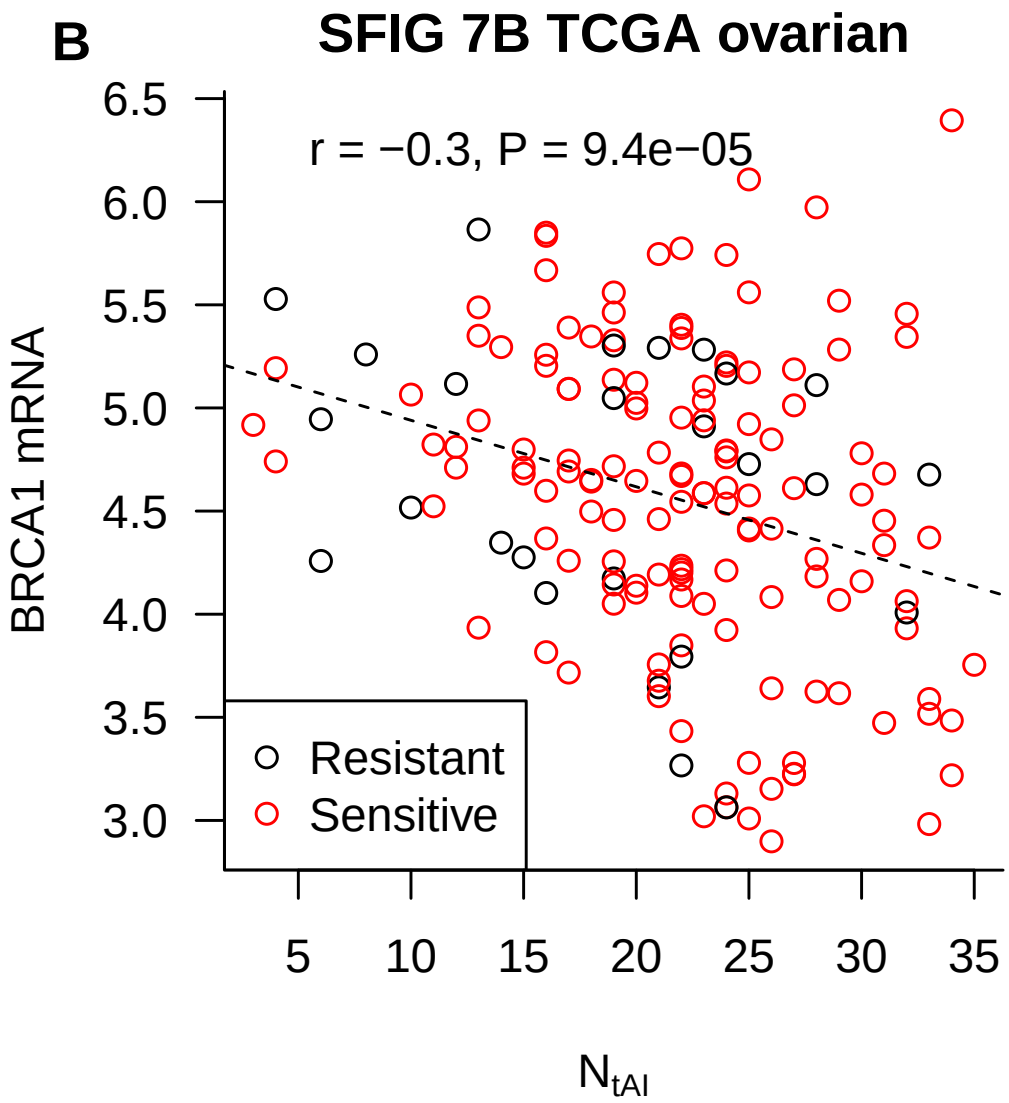
```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> plot(no.tel.tcga.tnbc[tcga_breast_tnbc.cont.sw >= cont.max, 1],
+     tcga.breast.ge.level3["BRCA1", tcga_breast_tnbc.cont.sw >=
+         cont.max], ylab = "BRCA1 mRNA", xlab = expression(N[tAI]),
+     main = "SFIG 7A TCGA TNBC", ylim = c(-3.5, 0.2))
> abline(lm(tcga.breast.ge.level3["BRCA1", tcga_breast_tnbc.cont.sw >=
+     cont.max] ~ no.tel.tcga.tnbc[tcga_breast_tnbc.cont.sw >=
+     cont.max, 1]), lty = 2)
> a <- cor.test(no.tel.tcga.tnbc[tcga_breast_tnbc.cont.sw >= cont.max,
+     1], tcga.breast.ge.level3["BRCA1", tcga_breast_tnbc.cont.sw >=
+     cont.max], method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> label.panel("A", xoff = 2)
```



Now in the ovarian cohort. Here we will subset the samples to only include those that have follow up data, and received platinum.

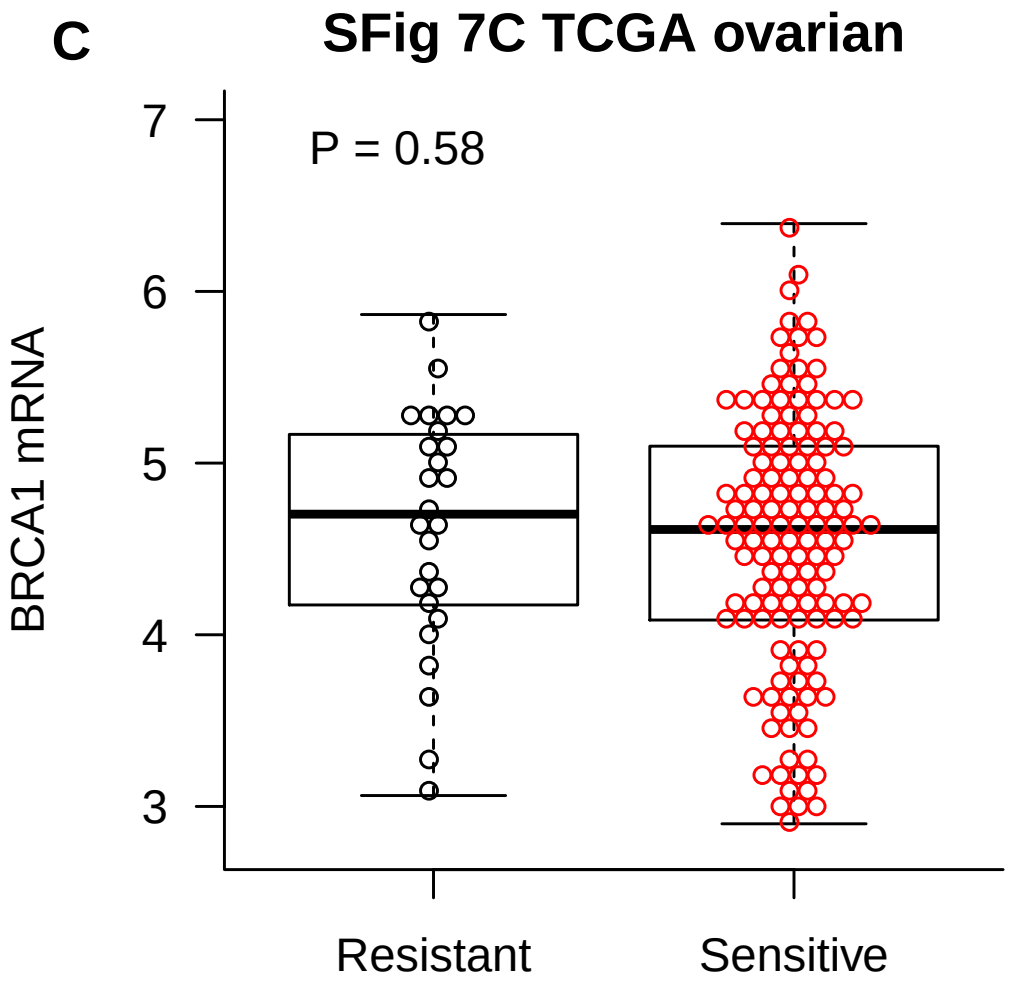
```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> tcga.subset.vec <- intersect(names(tcga.ov.wtbrca[tcga.ov.wtbrca]),
+     colnames(tcga.ovarian.eset))
> plot(no.tel.tcga[tcga.subset.vec, 1], tcga.ovarian.eset[brca.probe,
+     tcga.subset.vec], ylab = "BRCA1 mRNA", xlab = expression(N[tAI]),
+     col = c(1, 2)[match(tcga.ov.outcome.180[tcga.subset.vec],
+         c("FALSE", "TRUE"))], main = "SFIG 7B TCGA ovarian")
> abline(lm(tcga.ovarian.eset[brca.probe, tcga.subset.vec] ~ no.tel.tcga[tcga.subset.vec,
+     1]), lty = 2)
> a <- cor.test(no.tel.tcga[tcga.subset.vec, 1], tcga.ovarian.eset[brca.probe,
+     tcga.subset.vec], method = "spearman")
> legend("topleft", legend = paste("r = ", signif(a$estimate, 2),
+     ", P = ", signif(a$p.value, 2), sep = ""), lty = -1, pch = -1,
+     bty = "n")
> legend("bottomleft", legend = c("Resistant", "Sensitive"), col = c(1,
```

```
+     2), pch = 1)
> label.panel("B", xoff = 2)
```



And finally we will plot figure S7C, which shows the association between BRCA1 expression and response in the TCGA ovarian cohort, using the same samples as above.

```
> par(las = 1, cex = 1, bty = "l", pty = "s", mar = c(4, 4, 2,
+     2) + 0.1)
> boxplot(tcga.ovarian.eset[brca.probe, tcga.subset.vec] ~ tcga.ov.outcome.180[tcga.subset.vec],
+     ylab = "BRCA1 mRNA", xlab = "", notch = F, names = c("Resistant",
+         "Sensitive"), main = "SFig 7C TCGA ovarian", outline = F,
+     ylim = c(2.8, 7))
> beeswarm(tcga.ovarian.eset[brca.probe, tcga.subset.vec] ~ tcga.ov.outcome.180[tcga.subset.vec],
+     col = c(1, 2), add = T, pch = 1, method = "hex", cex = 0.8)
> a <- signif(wilcox.test(tcga.ovarian.eset[brca.probe, tcga.subset.vec] ~
+     tcga.ov.outcome.180[tcga.subset.vec])$p.value, 2)
> legend("topleft", legend = paste("P =", a), lty = -1, pch = -1,
+     bty = "n")
> label.panel("C", xoff = 2)
```


C
SFig 7C TCGA ovarian
P = 0.58
BRCA1 mRNA
7
6
5
4
3
Resistant
Sensitive

8 Optimize for platform-specific probe numbers

Four patient samples were tested on both versions of the Oncoscan platform: P11, P23, P24 and P27. The probe densities on two Oncoscan platforms were 42,000 probes and 330,000 probes respectively, giving an approximate ratio of 1:8.

Load data:

```
> load("cis1mcp.opto.RData")
> load("cis2mcp.opto.RData")
> load("cis1cna.opto.RData")
> load("cis2cna.opto.RData")
```

Test the ranks of telomeric and interstitial AI and CNA, when using no minimum number of probes

```
> mip1tel.tmp <- no.tel(cis1mcp.opto, chrominfo = chrominfo, min.size = 0,
+     min.probes = 1)
> mip2tel.tmp <- no.tel(cis2mcp.opto, chrominfo = chrominfo, min.size = 0,
+     min.probes = 1)
> mip1tel.cna.tmp <- no.tel.cna(cis1cna.opto, chrominfo = chrominfo,
+     min.size = 0, gain = log2(2.5/2), loss = log2(1.5/2), min.probes = 1)
> mip2tel.cna.tmp <- no.tel.cna(cis2cna.opto, chrominfo = chrominfo,
+     min.size = 0, gain = log2(2.5/2), loss = log2(1.5/2), min.probes = 1)
> mip1gaps <- gaps(cis1cna.opto, min.size = 0, gain = log2(2.5/2),
+     loss = log2(1.5/2), min.probes = 1)
> mip2gaps <- gaps(cis2cna.opto, min.size = 0, gain = log2(2.5/2),
+     loss = log2(1.5/2), min.probes = 1)
```

Perform test on ranks

```
> opto.1p <- vector()
> opto.1p[1] <- cor(mip1tel.tmp[, 1], mip2tel.tmp[, 1], method = "spearman")
> opto.1p[2] <- cor(mip1tel.tmp[, 3], mip2tel.tmp[, 3], method = "spearman")
> opto.1p[3] <- cor(mip1tel.cna.tmp[, 3], mip2tel.cna.tmp[, 3],
+     method = "spearman")
> opto.1p[4] <- cor((mip1gaps[, 5] - mip1tel.cna.tmp[, 3]), (mip2gaps[,
+     5] - mip2tel.cna.tmp[, 3]), method = "spearman")
> opto.1p[5] <- cor(mip1tel.cna.tmp[, 5], mip2tel.cna.tmp[, 5],
+     method = "spearman")
> opto.1p[6] <- cor((mip1gaps[, 6] - mip1tel.cna.tmp[, 5]), (mip2gaps[,
+     6] - mip2tel.cna.tmp[, 5]), method = "spearman")
> names(opto.1p) <- c("NtAI", "NiAI", "NtGain", "NiGain", "NtLoss",
+     "NiLoss")
> opto.1p

     NtAI      NiAI    NtGain    NiGain    NtLoss    NiLoss
0.9486833 0.3162278 0.8333333 0.4000000 0.9486833 0.8000000

> sum(opto.1p)

[1] 4.246928
```

Now test if the ranks are improved by using different numbers of proportional probes:

```
> cis.opto <- matrix(NA, nrow = 10, ncol = 6)
> colnames(cis.opto) <- c("NtAI", "NiAI", "NtGain", "NiGain", "NtLoss",
+     "NiLoss")
> count <- 0
> for (i in seq(5, 50, by = 5)) {
+     cat(i)
+     count <- count + 1
+     mip1tel.tmp <- no.tel(cis1mcp.opto, chrominfo = chrominfo,
+         min.size = 0, min.probes = i)
+     mip2tel.tmp <- no.tel(cis2mcp.opto, chrominfo = chrominfo,
+         min.size = 0, min.probes = (i * 8))
+     mip1tel.cna.tmp <- no.tel.cna(cis1cna.opto, chrominfo = chrominfo,
+         min.size = 0, gain = log2(2.5/2), loss = log2(1.5/2),
+         min.probes = i)
+     mip2tel.cna.tmp <- no.tel.cna(cis2cna.opto, chrominfo = chrominfo,
+         min.size = 0, gain = log2(2.5/2), loss = log2(1.5/2),
+         min.probes = (i * 8))
+     mip1gaps <- gaps(cis1cna.opto, min.size = 0, gain = log2(2.5/2),
+         loss = log2(1.5/2), min.probes = i)
+     mip2gaps <- gaps(cis2cna.opto, min.size = 0, gain = log2(2.5/2),
+         loss = log2(1.5/2), min.probes = (i * 8))
+     cis.opto[count, 1] <- cor(mip1tel.tmp[, 1], mip2tel.tmp[,
+         1], method = "spearman")
+     cis.opto[count, 2] <- cor(mip1tel.tmp[, 3], mip2tel.tmp[,
+         3], method = "spearman")
+     cis.opto[count, 3] <- cor(mip1tel.cna.tmp[, 3], mip2tel.cna.tmp[,
+         3], method = "spearman")
+     cis.opto[count, 4] <- cor((mip1gaps[, 5] - mip1tel.cna.tmp[,
+         3]), (mip2gaps[, 5] - mip2tel.cna.tmp[, 3]), method = "spearman")
+     cis.opto[count, 5] <- cor(mip1tel.cna.tmp[, 5], mip2tel.cna.tmp[,
+         5], method = "spearman")
+     cis.opto[count, 6] <- cor((mip1gaps[, 6] - mip1tel.cna.tmp[,
+         5]), (mip2gaps[, 6] - mip2tel.cna.tmp[, 5]), method = "spearman")
+ }
```

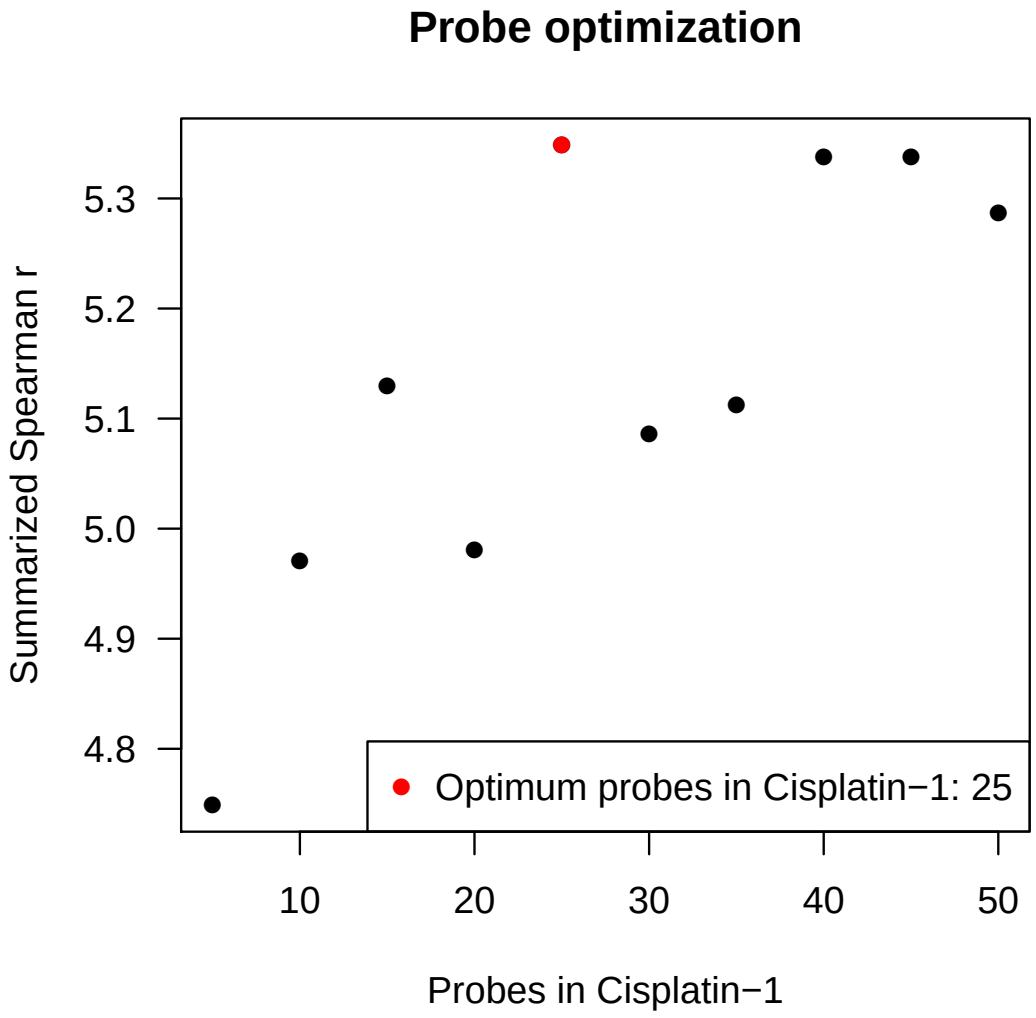
```
5101520253035404550
```

```
> rownames(cis.opto) <- seq(5, 50, by = 5)
> cis.opto <- cbind(cis.opto, apply(cis.opto, 1, sum))
> colnames(cis.opto)[7] <- "Sum"
> cis.opto
```

```
        NtAI      NiAI    NtGain NiGain    NtLoss    NiLoss      Sum
5  0.9486833 0.6000000 0.4000000    0.8 1.0000000 1.0000000 4.748683
10 0.8000000 0.7378648 0.6324555    0.8 1.0000000 1.0000000 4.970320
15 0.8000000 0.9486833 0.6324555    0.8 0.9486833 1.0000000 5.129822
20 0.8000000 0.9486833 0.6324555    0.8 0.8000000 1.0000000 4.981139
25 0.8000000 0.9486833 1.0000000    0.8 0.8000000 1.0000000 5.348683
30 0.8000000 0.7378648 0.9486833    0.8 0.8000000 1.0000000 5.086548
35 0.8000000 0.7378648 0.7745967    1.0 0.8000000 1.0000000 5.112461
40 0.8000000 0.7378648 1.0000000    1.0 0.8000000 1.0000000 5.337865
45 0.8000000 0.7378648 1.0000000    1.0 0.8000000 1.0000000 5.337865
50 0.8000000 0.7378648 1.0000000    1.0 0.8000000 0.9486833 5.286548
```

A proportional number of probes, starting at 25 probes in cisplatin-1 seems to show the greatest consistency

```
> par(las = 1)
> plot(rownames(cis.opto), cis.opto[, 7], pch = 16, main = "Probe optimization",
+     ylab = "Summarized Spearman r", xlab = "Probes in Cisplatin-1")
> points(rownames(cis.opto)[which(cis.opto[, 7] == max(cis.opto[,
+     7]))], max(cis.opto[, 7]), col = 2, pch = 16)
> legend("bottomright", legend = paste("Optimum probes in Cisplatin-1:",
+     rownames(cis.opto)[which(cis.opto[, 7] == max(cis.opto[,
+         7]))]), pch = 16, col = 2)
```



Using our technical replicates we determined the minimum probe number that gives the highest concordance of interstitial and telomeric AI/CNA calls by using proportional probe numbers.

We assumed that both platforms should detect the same pattern of genomic aberrations and observed differences are due to the differential probe coverage of the genome on the two platforms, which are corrected by the appropriate selection of minimum probe number.

9 Session info

The results in this file are generated using the following packages:

```
> sessionInfo()

R version 2.13.2 (2011-09-30)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

locale:
[1] C

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] boot_1.3-2         jetset_0.99.3      RSQLite_0.9-4      DBI_0.2-5
 [5] cluster_1.14.0     quantsmooth_1.18.0 lodplot_1.1        quantreg_4.69
 [9] SparseM_0.89       DNAcopy_1.26.0     ROC_1.28.0         gplots_2.8.0
[13] caTools_1.12       bitops_1.0-4.1     gdata_2.8.1        gtools_2.6.2
[17] beeswarm_0.1.1     genefilter_1.34.0  squash_0.3.1       affy_1.30.0
[21] Biobase_2.12.1

loaded via a namespace (and not attached):
[1] AnnotationDbi_1.14.1  affyio_1.20.0         annotate_1.30.0
[4] org.Hs.eg.db_2.5.0    preprocessCore_1.14.0 splines_2.13.2
[7] survival_2.36-9       tools_2.13.2          xtable_1.5-6

> system("uname -a", intern = TRUE)

[1] "Darwin njuul 11.2.0 Darwin Kernel Version 11.2.0: Tue Aug  9 20:54:00 PDT 2011; root:xnu-169
```