In [2]:
```
library(Homo.sapiens)
library(taRifx) ## Removes factors
library(scales)
library(SchramekLOH)
library(gplots)
library(IdeoViz)
library(reshape)
```

In [3]:
```
#detach("package:SchramekLOH", unload=TRUE)
#library(SchramekLOH)
```

Setup

Loading in all the precomputed data files

In [4]:
```
data("birdseed")  # df.bs, mapping.cov, mapping, ref.probe.ord
data("expr.mapping")  # expr.mapping
data("gaf")       # gaf
data("geneExpr")  # df.ex
data("mapping")   # mapping
data("snp6")      # snp6
data("Affyseg")   # affyseg
data("TCGAseg")   # seg
data("segmaf")    # segmaf
data("purity")    # purity
```

Load in the cbioportal obtained mutation attribute files

In [5]:
```
segdir <- '/Users/rquevedo/Onedrive/PughLab/Schramek_notch/IGV_segs/TCGA_hnsc'
```

In [6]:
```
mut.attr <- read.table(file.path(segdir, "hnsc_tcga_extended_attributes.ripk4.txt"
),
                       sep="\t", header=TRUE, stringsAsFactors = FALSE,
                       check.names = FALSE)
```

Setting up some of the paths

In [7]:
```
#### Load in Mappings ####
# Bed file obtained form http://www.affymetrix.com/Auth/analysis/downloads/lf/genotyping/GenomeWideSNP_6/GenomeWideSNP_6.hg19.bed.zip
pdir <- '/Users/rquevedo/Onedrive/PughLab/Schramek_notch/IGV_segs/TCGA_hnsc_vcf2'
#outdir <- '/Users/rquevedo/Onedrive/PughLab/Schramek_notch/IGV_segs/TCGA_hnsc_vcf2/summary_af'
outdir <- '/Users/rquevedo/Onedrive/PughLab/Schramek_notch/loh_analysis'
tmpdir <- file.path(outdir, "tmp")
plotsdir <- file.path(outdir, "plots")
dir.create(tmpdir, recursive = TRUE, showWarnings = FALSE)
dir.create(plotsdir, recursive = TRUE, showWarnings = FALSE)

ref <- file.path(pdir, "ref")


goi <- c("NOTCH1", "NOTCH2", "NOTCH3", "NOTCH4",
         "JAG1", "JAG2", "ADAM10", "AJUBA", "RIPK4")
```

Determine whether to use the TCGA Segs or the Affymetrix SNP6 Segs (same that the birdseeds originate from)

In [8]:
```
#### Read in Birdseed + Segs ####
use.affy <- FALSE
use.absolute <- TRUE
if(use.affy) seg <- affyseg
if(use.absolute) seg <- segmaf
```

Format the ABSOLUTE seg_maf file if using it

Takes the absolute files and maps the TCGA IDs on to the Affymetrix SNP6 IDs that were used. Also, includes the seg.mean as either the modal-CN values 0-centered at 2, or as the copy.ratio's 0-centered at 0.5

In [9]:
```
cn.center <- c('modal_cn', 'copy_ratio')
cn.center.choice <- 1
```

In [10]:
```
if(use.absolute){
    # Create mapping for Absoltue SEGMAF samples
    segmaf.ids <- unique(sort(seg$sample))
    segmaf.tmp <- data.frame("sample"=segmaf.ids[which(segmaf.ids %in% mapping$V3)
],
                             "ID"=gsub("01[AB]-.*", "01",
                                       mapIds(segmaf.ids[which(segmaf.ids %in% map
ping$V3)],
                                              mapping, in.type='affy', out.type='t
cga')))

    seg <- merge(seg, segmaf.tmp, by='sample', all.x=TRUE)
    if(cn.center[cn.center.choice] == 'modal_cn') seg$seg.mean <- (seg$modal_cn -
2)
    if(cn.center[cn.center.choice] == 'copy_ratio') seg$seg.mean <- (seg$copy_rati
o - 0.5)

    colnames(seg) <- c('sample', 'chrom','loc.start','loc.end','num.mark','length'
,
                       'seg_sigma','W','copy_ratio','modal_cn','expected_cn','subc
lonal',
                       'cancer_cell_frac','ccf_ci95_low','ccf_ci95_high','hz','ID'
, 'seg.mean')
}
```

Format the IDs for the purity table calculated for ABSOLUTE

In [11]:
```
if(use.absolute){
  purity.ids <- purity$sample
  purity.tmp <- data.frame("sample"=purity.ids[which(purity.ids %in% mapping$V3)],
                           "ID"=gsub("01[AB]-.*", "01",
                                     mapIds(purity.ids[which(purity.ids %in% mappi
ng$V3)],
                                            mapping, in.type='affy', out.type='tcg
a')))
  purity <- merge(purity, purity.tmp, by="sample", all.x=TRUE)
}
```

Preprocess

Ordering all the data structures

In [12]:
```
seg$chrom <- gsub("(chr).*\\1", "\\1", paste0("chr", seg$chrom))
seg.ids <- split(seg, f=seg$ID)
seg.i <- seg.ids[[1]]
head(seg.i)
```

	sample	chrom	loc.start
302691	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr10	51595847
302692	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr1	106216780
302693	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr10	46943377
302694	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr22	21721603
302695	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr10	51231576
302696	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr7	149968236

In [13]:
```
#### Chromosome order datasets ####
seg$chrom <- gsub("(chr).*\\1", "\\1", paste0("chr", seg$chrom))
seg.ids <- split(seg, f=seg$ID)
seg.chr <- lapply(seg.ids, function(seg.i){
  seg.i <- seg.i[order(seg.i$loc.start),]
  seg.tmp <- split(seg.i, f=seg.i$chrom)

  chr.ord <- paste0("chr", c(1:22, "X", "Y"))
  x.factor <- factor(names(seg.tmp), levels = chr.ord, ordered=TRUE)

  seg.tmp[order(x.factor)]
})
```

In [14]:
```
if(use.absolute)  {
    purity.ids <- split(purity, f=purity$ID)
    print(head(names(purity.ids)))
    head(purity.ids[[1]])
}
```

```
[1] "TCGA-4P-AA8J-01" "TCGA-BA-4074-01" "TCGA-BA-4075-01" "TCGA-BA-4076-01"
[5] "TCGA-BA-4077-01" "TCGA-BA-4078-01"
```

	sample	array
685	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	MESNE_p_TCGAb_401_

In [15]:
```
head(seg.chr[['BALMS_p_TCGAb54and67_SNP_N_GenomeWideSNP_6_A03_730402']][['chr3']])
head(seg.chr[['TCGA-4P-AA8J-01']][['chr3']])
```

```
NULL
```

	sample	chrom	loc.start
303281	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr3	60345
303212	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr3	93540792
303132	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr3	97956985
303215	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr3	97970068
303340	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr3	114660592
303359	MESNE_p_TCGAb_401_02_03_04_05_N_GenomeWideSNP_6_D01_1486852	chr3	114668950

In [16]:
```
#### Map Probesets to Genomic Loci ####
if(exists("ref.probe.ord")){
    snp6 <- snp6[match(ref.probe.ord, snp6$V4),]
    snp6.ord <- snp6[,c(4, 1:3)]
    colnames(snp6.ord) <- c("probeset_id", "chrom", "start", "end")

    snp6.chr <- split(snp6.ord, f=snp6.ord$chrom)
    bs.chr <- split(as.data.frame(df.bs), snp6.ord$chrom)
    goi.df <- getGeneLoci(goi)
    goi.chr <- split(goi.df, f=goi.df$chr)

    chrom.ord <- match(paste0("chr", c(1:22, "X", "Y")), names(snp6.chr))
    snp6.chr <- snp6.chr[chrom.ord]
    bs.chr <- bs.chr[chrom.ord]
}
```

Formatting the Gene expression data

In [17]:
```
#### Expression analysis
if(exists("df.ex")){
    ## Generate z-score per gene
    z <- function(x){ (x - mean(x, na.rm=TRUE)) / sd(x, na.rm=TRUE)}
    z.ex <- data.frame(t(apply(df.ex, 1, z)), stringsAsFactors=FALSE)

    ## Map Genes to Genomic Loci ##
    ord <- match(rownames(df.ex), gaf$V2)
    gaf.ord <- gaf[ord, c("V2", "V17")]
    gaf.ord$chr <- gsub(":.*", "", gaf.ord$V17)
    gaf.ord$start <- as.numeric(gsub("^.*:", "", gsub("-.*", "", gaf.ord$V17)))
    gaf.ord$end <- as.numeric(gsub(":.*", "", gsub("^.*?-", "", gaf.ord$V17)))

    ## Reorder all the matrices into genomic loci numerical order
    chr.ord <- paste0("chr", c(1:22, "X", "Y"))
    gaf.ord <- gaf.ord[order(gaf.ord$start),]
    gaf.ord <- gaf.ord[order(match(gaf.ord$chr, chr.ord)), ]

    ord <- match(gaf.ord$V2, rownames(df.ex))
    df.ex <- df.ex[ord,]
    z.ex <- z.ex[ord,]

    ## Order the list by chromosomes
    gaf.chr <- split(gaf.ord, f=gaf.ord$chr)
    z.chr <- split(z.ex, f=gaf.ord$chr)

    chrom.ord <- match(paste0("chr", c(1:22, "X", "Y")), names(gaf.chr))
    gaf.chr <- gaf.chr[chrom.ord]
    z.chr <- z.chr[chrom.ord]
}
```

Comparison of overlapping Expression and Seg arrays

In [18]:
```
print(paste0("Number of Affy6 samples: ", length(names(seg.ids))))
print(paste0("Number of Affy6 samples: ", length(names(z.ex))))
print(paste0("Number of Affy6 samples: ", length(names(seg.ids))))

length(intersect(names(seg.ids), gsub("\\.", "-", names(z.ex))))
```

```
[1] "Number of Affy6 samples: 497"
[1] "Number of Affy6 samples: 506"
[1] "Number of Affy6 samples: 497"
```

497

Analysis

Visualization and generate StdRes

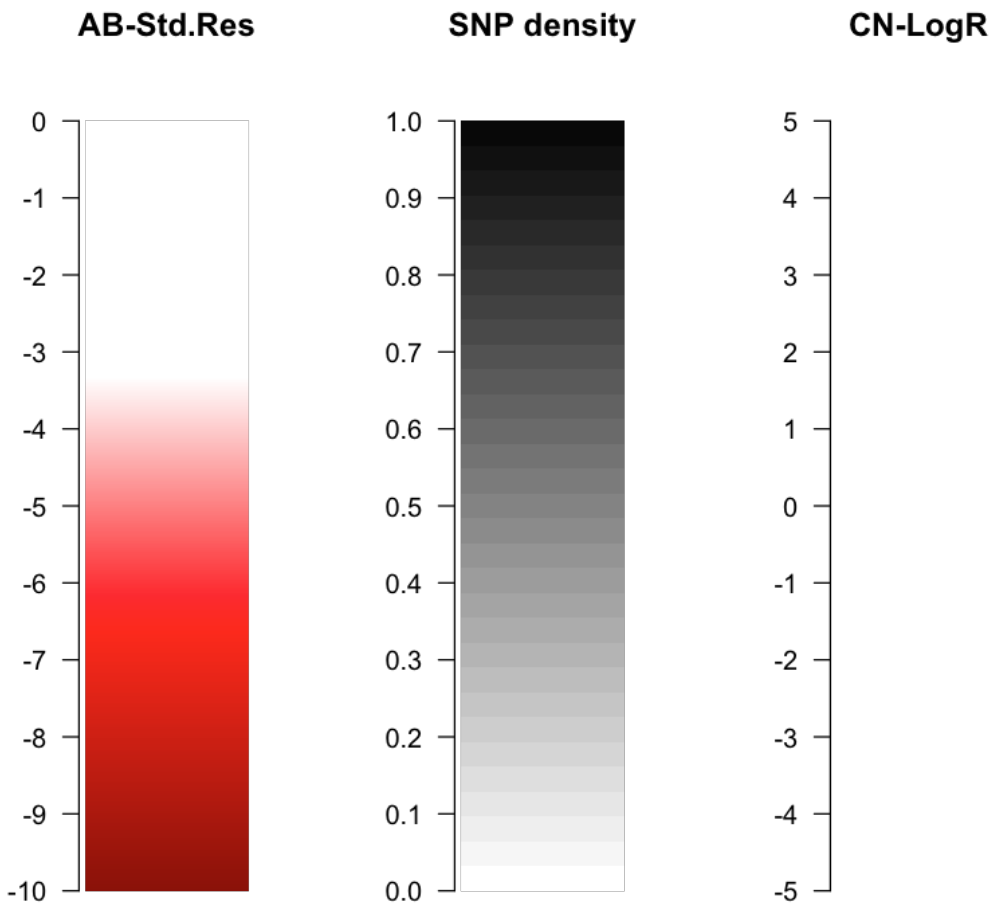
In [19]:
```
library(gplots)
library(RColorBrewer)
```

Setting up the colours that will be used for all visualizations

In [20]:
```
rf <- colorRampPalette(c("white", "black"))
pf <- colorRampPalette(c("white", "white", "red", "darkred"))
cf <- colorRampPalette(c("blue4", "blue4", "blue", "white", "red", "darkred",  "da
rkred"))
r <- rf(32); p <- pf(1000); cn <- cf(100)
names(cn) <- round(seq(-4.9, 5.0, by=0.1),1)
```

Visualization for the colour bars and ranges used

In [511]:
```
#pdf(file.path(plotsdir, "legend.pdf"), width=6)
null <- split.screen(c(1,3))
screen(1); color.bar(p, min=0, max=-10, title="AB-Std.Res")
screen(2); color.bar(r, min=0, max=1, title="SNP density")
screen(3); color.bar(cn, min=-5, max=5, title="CN-LogR")
close.screen(all.screens=TRUE)
#dev.off()
```



In [22]:
```
bin.size <- 1000000
spacer.param <- 20
```

In [1534]:
```
sample.stdres <- suppressWarnings(lapply(colnames(bs.chr[[1]]), mapAndPlotFeature
s,
                          mapping.cov=mapping.cov, use.aff=use.affy,
                          plotsdir=plotsdir, snp6.chr=snp6.chr,
                          seg.chr=seg.chr, bs.chr=bs.chr, z.chr=z.chr,
                          r=r, p=p, cn=cn, gen.plot=TRUE, use.absolute=use.absolute
))
```

Summarize the standardized residuals data into data frames

In [1535]:
```
all.stdres <- lapply(sample.stdres, function(i) i[['all']])
names(all.stdres) <- colnames(bs.chr[[1]])
## Reduce the gene to a single segment
sample.stdres.bkup <- sample.stdres
sample.stdres <- lapply(sample.stdres, function(i) {
  single.j <- sapply(split(i[['genes']], f=i[['genes']]$gene), function(j){
    uniq.j <- apply(j, 2, unique)
    if(any(sapply(uniq.j[c('seg.start', 'seg.end', 'seg.mean')], length) > 1)){
      uniq.j[['seg.start']] <- min(uniq.j[['seg.start']])
      uniq.j[['seg.end']] <- max(uniq.j[['seg.end']])
      uniq.j[['seg.mean']] <- mean(uniq.j[['seg.mean']])
    }

    sapply(uniq.j, function(x) x)
  })
  remove.factors(data.frame(t(single.j)))
})
names(sample.stdres) <- colnames(bs.chr[[1]])
```

In [1537]:
```
head(all.stdres[['FISTS_p_TCGA_b107_121_SNP_N_GenomeWideSNP_6_D08_777884']],3)
```

	chrom	start	end	stdres
[0,1e+06]	chr1	0	1000797	-2.5980762
(1e+06,2e+06]	chr1	1000797	2001595	-0.9878292
(2e+06,3e+06]	chr1	2001595	3002393	-0.3333333

In [1538]:
```
head(sample.stdres[['FISTS_p_TCGA_b107_121_SNP_N_GenomeWideSNP_6_D08_777884']],3)
```

	gene	chr	gene.start	gene.end	bin.start	bin.end	seg.start	seg.end	seg.mean
ADAM10	ADAM10	chr15	58888510	59042177	58814431	59621572	34794922	74252408	0
AJUBA	AJUBA	chr14	23440410	23451848	22652964	23466738	19382209	35724239	-1
JAG1	JAG1	chr20	10618332	10654694	10038090	11035720	69408	29529825	-1

In [ ]:
```
#save(all.stdres, sample.stdres, sample.stdres.bkup, file=file.path(tmpdir, paste0
("tmp", use.affy, ".RData")))
#load(file.path(tmpdir, paste0("tmp", use.affy, ".RData")))
```

In [23]:
```
#save(all.stdres, sample.stdres, sample.stdres.bkup, file=file.path(tmpdir, paste0
("tmpABSOLUTE", use.affy, ".RData")))
load(file.path(tmpdir, paste0("tmpABSOLUTE", use.affy, ".RData")))
```

Plot the SEGMAF files

This generates copy-number profiles of copy-ratio ont he y-axis, coloured by copy-state

In [ ]:
```
if(use.absolute){
    seg.ids <- split(segmaf, f=segmaf$sample)
    lapply(seg.ids, plotSeg, outdir=outdir)
}
```

Generate 'Attributes' files for use in IGV

Summarize all the Standardized Residuals into LOH/Het value annotations for IGV visualization

In [24]:
```
stdres.thresh <- -5
attributes <- lapply(seq_along(sample.stdres), generateIgvAttributes,
                     sample.stdres=sample.stdres, mapping.cov=mapping.cov,
                     stdres.thresh=-5)
attributes <- do.call("rbind", attributes)
head(attributes, 5)
```

	TRACK_ID	ADAM10	AJUBA	JAG1	JAG2	NOTCH1	NOTCH2	NOTCH3	NOTCH4	RIPK4
loh.val	TCGA-CN-6011-01	Het	Het	Het	LOH	Het	LOH	Het	Het	LOH
loh.val1	TCGA-CN-6012-01	Het	LOH	LOH	LOH	LOH	LOH	Het	Het	LOH
loh.val2	TCGA-CN-6016-01	Het	Het	LOH	LOH	LOH	Het	LOH	LOH	Het
loh.val3	TCGA-CN-6018-01	Het	Het	LOH	LOH	LOH	Het	Het	LOH	LOH
loh.val4	TCGA-CN-6019-01	LOH	LOH	LOH	LOH	Het	Het	LOH	LOH	LOH

Write and save the data structures

In [1540]:
```
write.table(attributes, file.path(outdir, "LOH_attributes.txt"),
            sep="\t", quote=FALSE, col.names=TRUE, row.names=FALSE)
save(sample.stdres, attributes, mapping.cov, file=file.path(outdir, "gene_stdRes.
R"))
```

Generate contigency tables and test for significance

Initialize the contigency table to be used for quick reference later

In [1966]:
```
all.ctbl <- doTheChi(ctbl=NULL, attributes, mut.attr,
                     tbl.idx=c(1,3), gene='ADAM10', mut='ADAM10_CNA')
ctbl <- all.ctbl[['ctbl']]
```

Available mutations to compare for LOH, where "ADAM10" actually means "ADAM10_LOH"

In [1967]:
```
print(names(ctbl))
```

```
 [1] "NOTCH1_CNA"    "NOTCH1_MUT"    "NOTCH1_FUSION" "NOTCH2_CNA"
 [5] "NOTCH2_MUT"    "NOTCH2_FUSION" "NOTCH3_CNA"    "NOTCH3_MUT"
 [9] "NOTCH3_FUSION" "NOTCH4_CNA"    "NOTCH4_MUT"    "NOTCH4_FUSION"
[13] "JAG1_CNA"      "JAG1_MUT"      "JAG1_FUSION"   "JAG2_CNA"
[17] "JAG2_MUT"      "JAG2_FUSION"   "RIPK4_CNA"     "RIPK4_MUT"
[21] "RIPK4_FUSION"  "ADAM10_CNA"    "ADAM10_MUT"    "ADAM10_FUSION"
[25] "AJUBA_CNA"     "AJUBA_MUT"     "AJUBA_FUSION"  "ADAM10"
[29] "AJUBA"         "JAG1"          "JAG2"          "NOTCH1"
[33] "NOTCH2"        "NOTCH3"        "NOTCH4"        "RIPK4"
```

Run the chi-squared analysis on the samples that are of interest

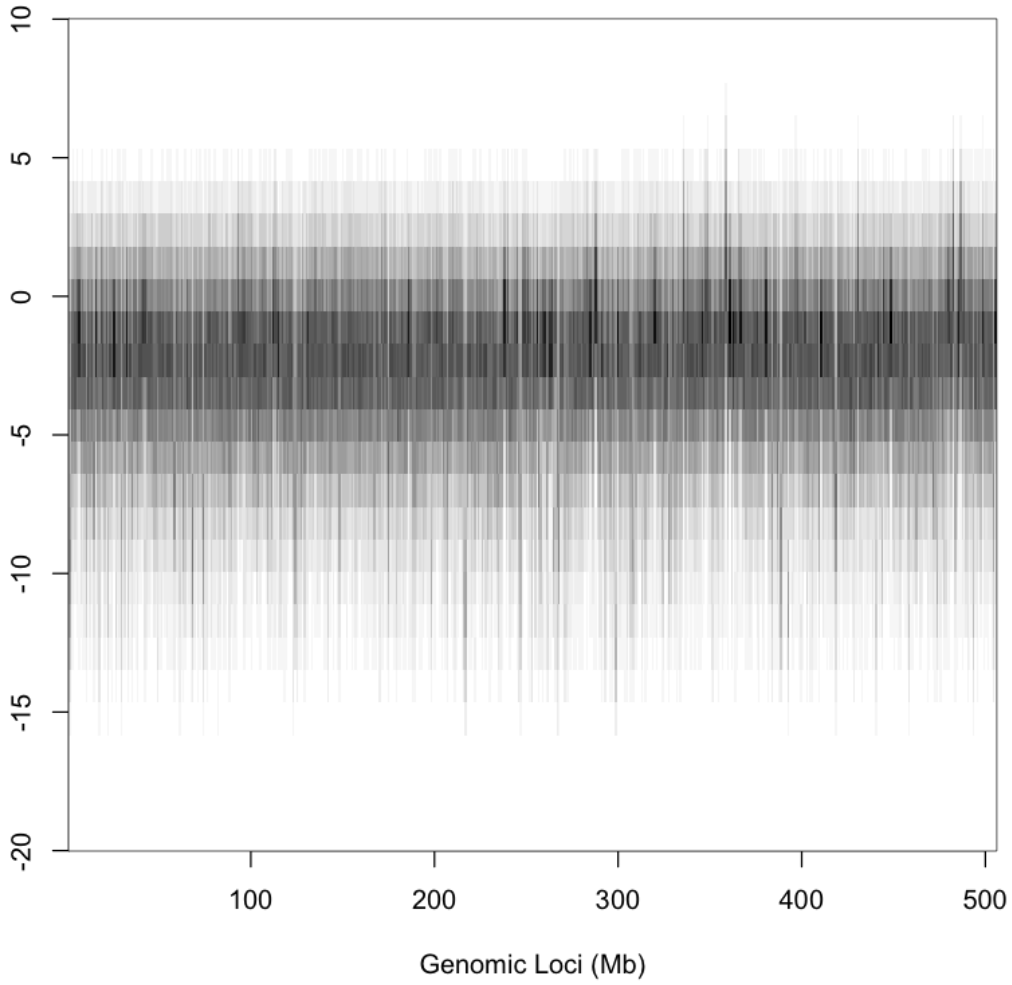
In [1968]:
```
print(doTheChi(ctbl=ctbl, tbl.idx=c(1,3), gene='ADAM10', mut='ADAM10_CNA'))
print(doTheChi(ctbl=ctbl, tbl.idx=c(1,2), gene='NOTCH1', mut='NOTCH1_CNA'))
print(doTheChi(ctbl=ctbl, tbl.idx=c(1,3), gene='AJUBA', mut='AJUBA_CNA'))
```

```
$Contigency
     j
i     HETLOSS HOMDEL no_alteration
  Het      53      0           183
  LOH      58      1           211

$p
[1] 0.8926257

$Std.Res
     j
i         HETLOSS no_alteration
  Het  0.2426701    -0.2426701
  LOH -0.2426701     0.2426701

$Contigency
     j
i     HOMDEL no_alteration
  Het      4           245
  LOH      7           250

$p
[1] 0.5777054

$Std.Res
     j
i          HOMDEL no_alteration
  Het -0.8616207     0.8616207
  LOH  0.8616207    -0.8616207

$Contigency
     j
i     HETLOSS HOMDEL no_alteration
  Het      24      1           126
  LOH      51      0           304

$p
[1] 0.7377338

$Std.Res
     j
i         HETLOSS no_alteration
  Het  0.4717808    -0.4717808
  LOH -0.4717808     0.4717808
```

StdRes plots

In [24]:
```
#pdf(file.path(outdir, "stdres.pdf"), width=20)
m.stdres <- do.call("rbind", all.stdres)
m.stdres$ID <- gsub("\\..*", "", rownames(m.stdres))
m.stdres <- melt(m.stdres, id.vars = "ID", measure.vars = "stdres")
m.stdres$IDidx <- match(m.stdres$ID, unique(m.stdres$ID))
h2 <- hist2d(m.stdres[,c("IDidx", "value")],
             nbins=c(length(unique(m.stdres$ID)), 30),
             col=r, ylim=c(-20,10),
             xlab = "Genomic Loci (Mb)")
#dev.off()
```

```
Attaching package: ‘reshape’

The following objects are masked from ‘package:S4Vectors’:

    expand, rename
```



Ideogram of HetLoss over Gene of Interest

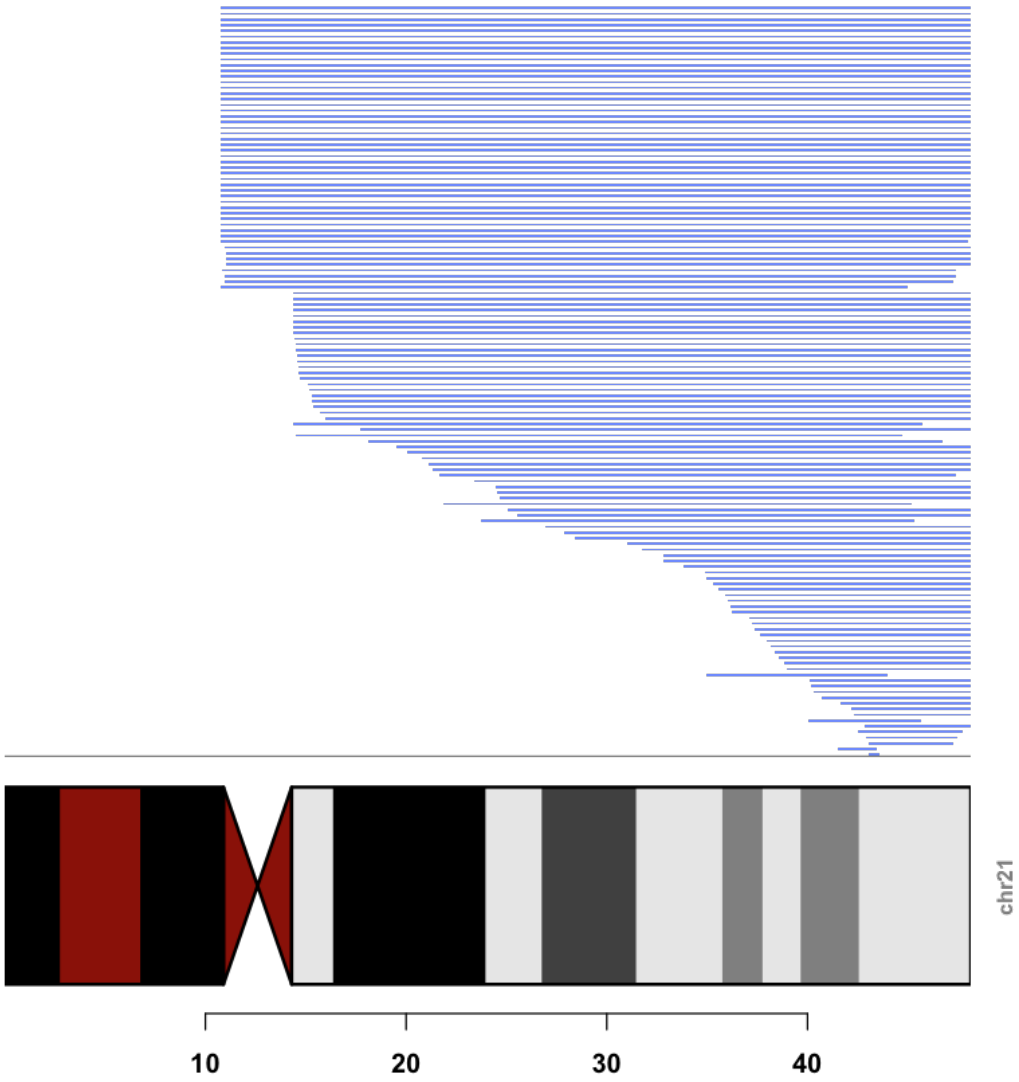
Set up the Ideogram hg19 reference

In [25]:
```
ideo_hg19 <- getIdeo("hg19")
```

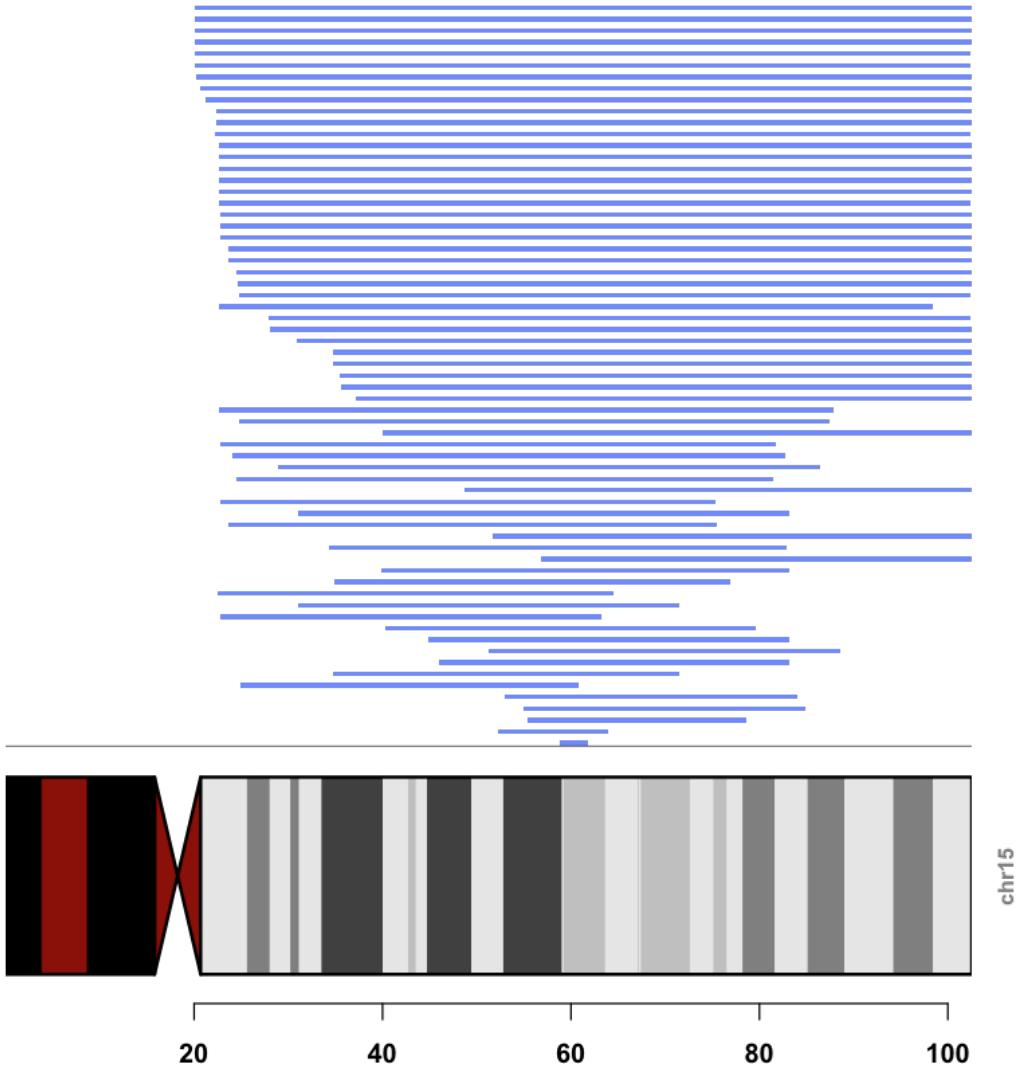
Visualize each gene of interest in their ideogram

In [30]:
```
for(gene in rownames(sample.stdres[[1]])){
  pdf(file.path(outdir, paste0("ideo_", gene, ".pdf")), height=20)
  plotIdeoGene(ideo_hg19, sample.stdres, gene, thresh=-0.1)
  dev.off()
}
```

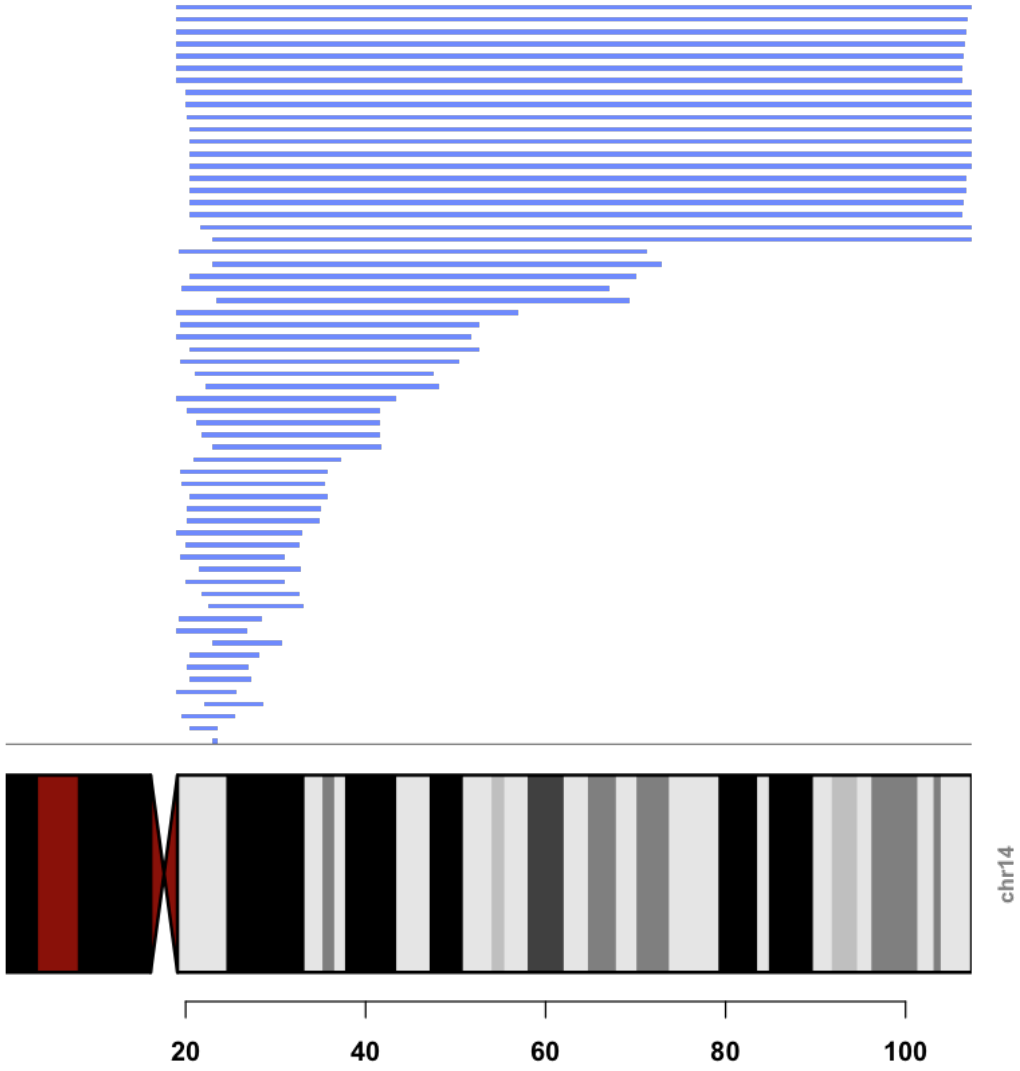
In [484]:
```
plotIdeoGene(ideo_hg19, sample.stdres, 'RIPK4', thresh=0)
```



In [1971]:
```
plotIdeoGene(ideo_hg19, sample.stdres, 'ADAM10', thresh=0)
```



```
In [1972]: plotIdeoGene(ideo_hg19, sample.stdres, 'AJUBA', thresh=0)
```



Chi Square of Absolute CN calls

This is meant to use the ABSOLUTE called seg_maf files. A LOSS or HETLOSS is called using the absolute copy number (0-centered modal CN) where the modal_cn is less than 0. Based on these calls, it generates a contigency table of LOSS events, as well as flags the overlapping samples. IT also explores to some degree on why those cases are overlapping

In [26]:
```
c.tbl <- function(g1, g2){
    x <- lapply(names(sample.stdres), function(x.id){
        x <- sample.stdres[[x.id]]
        if(x[g1, 'seg.mean'] < 0) ad <- paste(g1, '_LOSS') else ad <- 'NA'
        if(x[g2, 'seg.mean'] < 0) aj <- paste(g2, '_LOSS') else aj <- 'NA'
        if(grepl('LOSS', ad) & grepl('LOSS', aj)) {
            id.df <- data.frame('affy'=x.id,
                                'tcga'=gsub("01[AB]-.*", "01",
                                            mapIds(x.id, mapping.cov,
                                                   in.type='affy', out.type='tcga'
)))
        } else { id.df <- NULL }
        list("cnt"=c(ad, aj), "ids"=id.df)
    })

    z <- sapply(x, function(y) y[['cnt']])
    id.df <- do.call("rbind", lapply(x, function(y) y[['ids']]))
    list("tbl"=table(t(z)[,1], t(z)[,2]),
         "ids"=id.df)
}
```

In [318]:
```
x <- c.adam.aj[['tbl']]
x

log2((x[1,1] * x[2,2]) / (x[1,2] * x[2,1]))
```

```
              AJUBA _LOSS  NA
  ADAM10 _LOSS           9  56
  NA                    52 389
```

0.265756706229808

In [59]:
```
#if(use.absolute){
    c.adam.aj <- c.tbl('ADAM10', 'AJUBA'); print(c.adam.aj)
    c.adam.rip <- c.tbl('ADAM10', 'RIPK4'); print(c.adam.rip[['tbl']])
    c.aj.rip <- c.tbl('AJUBA', 'RIPK4'); print(c.aj.rip[['tbl']])
#}
```

```
$tbl

              AJUBA _LOSS  NA
  ADAM10 _LOSS           9  56
  NA                    52 389

$ids
                                                              affy            tcga
1       CLUBS_p_TCGA_186_188_SNP_N_GenomeWideSNP_6_B05_913998 TCGA-CQ-6221-01
2       MAULS_p_TCGA_189_190_SNP_N_GenomeWideSNP_6_C11_932832 TCGA-CV-7432-01
3      PALPS_p_TCGA_265_266_267_N_GenomeWideSNP_6_A02_1306818 TCGA-CV-A460-01
4  RICES_p_TCGA_Batch_310_311_NSP_GenomeWideSNP_6_D03_1361892 TCGA-CQ-7072-01
5 UNDID_p_TCGA_353_354_355_37_NSP_GenomeWideSNP_6_G09_1376838 TCGA-H7-A76A-01
6 UNDID_p_TCGA_353_354_355_37_NSP_GenomeWideSNP_6_E09_1376946 TCGA-MT-A7BN-01
7     CUTCH_p_TCGAb_355_37_52_NSP_GenomeWideSNP_6_G09_1376812 TCGA-UF-A719-01
8  TYPED_p_TCGA_188_192_Mirn_SNP_N_GenomeWideSNP_6_C06_913818 TCGA-IQ-7632-01
9      PALPS_p_TCGA_265_266_267_N_GenomeWideSNP_6_A12_1306808 TCGA-CQ-A4C9-01


               NA RIPK4 _LOSS
  ADAM10 _LOSS  40          25
  NA           334         107

              NA RIPK4 _LOSS
  AJUBA _LOSS  37          24
  NA          337         108
```

Probing each case individually, I see that these are all relatively low purity samples (purity < 0.4). However, by manually inspecting CN plots, it still seems fairly ocnfident in calling the CN of those regions and I have no real reason to doubt those calls.

However, what I do see is that the AJUBA mutants are all VERY focal

In [28]:
```
.reportCases <- function(id){
    id <- gsub("01[AB]-.*", "01", mapIds(id, mapping.cov, in.type='affy', out.type
='tcga'))
    chr14 <- seg.chr[[id]][['chr14']]
    chr15 <- seg.chr[[id]][['chr15']]


    ajuba.idx <- which(with(chr14, loc.start < 23438410 & loc.end > 23453848))
    adam.idx <- which(with(chr15, loc.start < 58886510 & loc.end > 59044177))
    flank <- 1

    list("purity"=purity.ids[[id]][,c(1, 4,5,10 )],
         'ajuba'=chr14[c((ajuba.idx-flank):(ajuba.idx+flank)), c(17, 2:4, 6, 18)],
         'adam10'=chr15[c((adam.idx-flank):(adam.idx+flank)), c(17, 2:4, 6, 18)])
}
```

In [61]:
```
#lapply(c.adam.aj[['ids']][['affy']], .reportCases)
```

While all of the samples have low purity, it seems like 3/4 (indicated by **) are interesting cases:

```
** 1       MAULS_p_TCGA_189_190_SNP_N_GenomeWideSNP_6_C11_932832 TCGA-CV-7432-01
** 2      PALPS_p_TCGA_265_266_267_N_GenomeWideSNP_6_A02_1306818 TCGA-CV-A460-01
** 3  RICES_p_TCGA_Batch_310_311_NSP_GenomeWideSNP_6_D03_1361892 TCGA-CQ-7072-01
4 UNDID_p_TCGA_353_354_355_37_NSP_GenomeWideSNP_6_G09_1376838 TCGA-H7-A76A-01
```

These 3 all have a HETLOSS of AJUBA, but these losses are within 1-2Mb of a neutral or a gain region. This suggests that they may be cases of mis-segmentation, or amplification of promoter/enhancer region to rectify haploinsufficiency.

An inspection of the expression of probes at that site should clarify these cases

Checking AJUBA expression of overlapping cases in context of purity

In [1613]:
```
gene <- 'AJUBA'
gene.alt <- 'JUB' # Alternate name of AJUBA
```

Using the overlapping cases, set the range of purities to build a reference expression distribution around

In [1614]:
```
overlapping.cases <- as.character(c.adam.aj[['ids']][['tcga']])
overlapping.purity <- sapply(purity.ids[overlapping.cases], function(x) x['purity
'])
purity.range <- c(min(as.numeric(overlapping.purity)),
                  max(as.numeric(overlapping.purity)))
print(purity.range)
```

```
[1] 0.29 0.98
```

Using the purity range just established, find all the samples that fall between these values

In [1615]:
```
valid.purity <- sapply(purity.ids, function(x) x['purity'] > min(purity.range) &
x['purity'] < max(purity.range))
valid.samples <- names(purity.ids)[which(unlist(valid.purity))]
print(head(valid.samples))
```

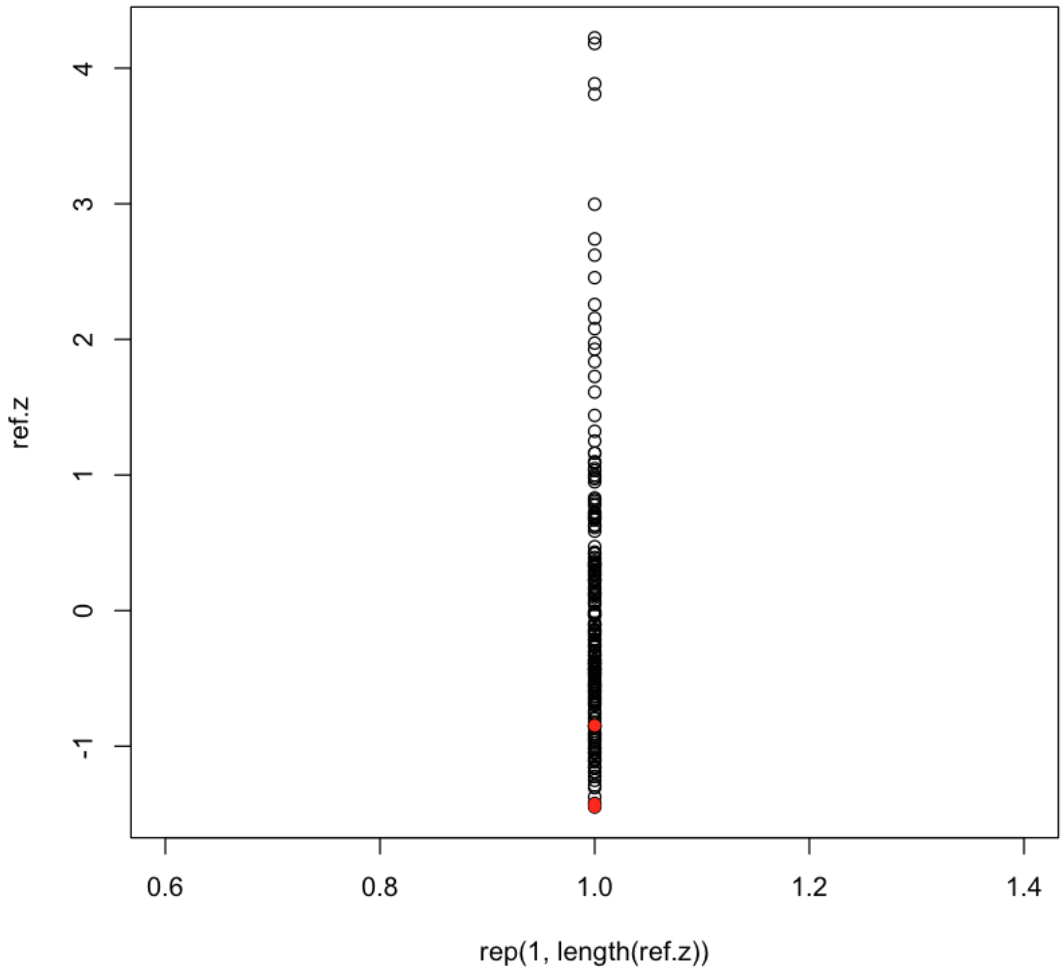
```
[1] "TCGA-4P-AA8J-01" "TCGA-BA-4074-01" "TCGA-BA-4075-01" "TCGA-BA-4076-01"
[5] "TCGA-BA-4077-01" "TCGA-BA-4078-01"
```

Using the valid samples that fall within the purity range, build a reference distribution to calculate a z score from

In [1616]:
```
ajuba.exp <- getGeneExp('JUB', df.ex)
ref.exp <- ajuba.exp[gsub("-", ".", valid.samples), 1, drop=FALSE]
ref.exp.num <- ref.exp[,1]
ref.z <- (ref.exp.num - mean(ref.exp.num, na.rm=TRUE)) / (sd(ref.exp.num, na.rm=T
RUE))
```

Identify the overlapping cases in the sea of the Z-scores under the purity-selected reference distributions

In [1617]:
```
plot(x=rep(1, length(ref.z)), ref.z)
points(x=rep(1, length(overlapping.cases)),
       ref.z[match(gsub("-", ".", overlapping.cases), rownames(ref.exp))],
       col='red', pch=16)
```



Unfortunately, only 1 of the 4 cases had corresponding expression data, so validating based on z-score of expression was not really plausible. That one case also had lower z-score than all other cases of ajuba that fell within expression

Expression of AJUBA and ADAM10 Mutants

AJUBA

Grab gene expression for all samples and assign them to their proper data structures.

Then, separate the samples based on their mutations

In [509]:
```
ex.by.mut <- parseIdsByMutation('AJUBA', getGeneExp('JUB', z.ex),
                                seg.ids=seg.ids, lo.q=0.1, hi.q=0.9,
                                cbio=cbio.abs, keep.only.abs=TRUE)
null <- .mutBoxplot(ex.by.mut, add.purity=FALSE)
lapply(ex.by.mut, head, 3)
```

$`NA`

TRACK_ID	Alt	Alt.type	JUB.84962	IQR.90.
TCGA-BA-4074-01	no_alteration	NA	-1.0212289	1
TCGA-BA-4076-01	no_alteration	NA	-0.1099163	2
TCGA-BA-4077-01	no_alteration	NA	-0.3082486	2

$mut

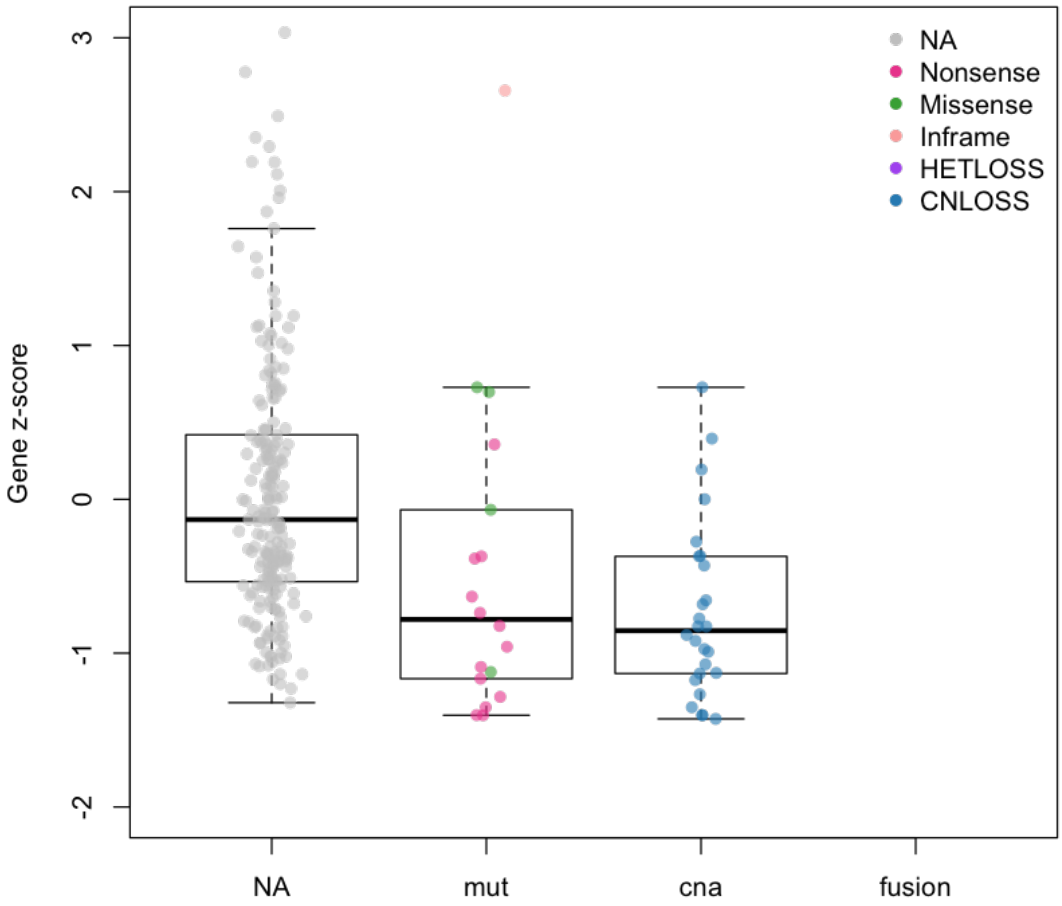
TRACK_ID	AJUBA_MUT	Alt.type	JUB.84962	IQR.90.
TCGA-BA-5556-01	R428Q,I304Dfs*2	Nonsense	-0.6325205	0
TCGA-BB-A5HY-01	S230Ffs*76	Nonsense	NA	2
TCGA-BB-A6UO-01	V264Lfs*2	Nonsense	NA	2

$cna

	TRACK_ID	AJUBA_CNA	Alt.type	JUB.84962	IQR.90.
1	TCGA-4P-AA8J-01	CNLOSS	CNLOSS	NA	1
3	TCGA-BA-4078-01	CNLOSS	CNLOSS	-0.6562828	1
6	TCGA-BA-6871-01	CNLOSS	CNLOSS	-0.8815935	2

$fusion

TRACK_ID	AJUBA_FUSION	Alt.type	JUB.84962	IQR.90.


NA
Nonsense
Missense
Inframe
HETLOSS
CNLOSS
Gene z-score
3
2
1
0
-1
-2
NA
mut
cna
fusion

In [510]:
```
t.test(ex.by.mut[['cna']][,4], ex.by.mut[['NA']][,4])
t.test(ex.by.mut[['mut']][,4], ex.by.mut[['NA']][,4])
```

```
        Welch Two Sample t-test

data:  ex.by.mut[["cna"]][, 4] and ex.by.mut[["NA"]][, 4]
t = -6.3296, df = 45.13, p-value = 0.00000009991
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.0935714 -0.5656446
sample estimates:
  mean of x   mean of y
-0.73151307  0.09809489

        Welch Two Sample t-test

data:  ex.by.mut[["mut"]][, 4] and ex.by.mut[["NA"]][, 4]
t = -2.2136, df = 19.625, p-value = 0.03888
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.09293007 -0.03176499
sample estimates:
  mean of x   mean of y
-0.46425263  0.09809489
```

In [500]:
```
pdf(file.path(outdir, "ajuba_mut_expr.pdf"), width=5)
null <- .mutBoxplot(ex.by.mut, add.purity=FALSE, jitter=0.1)
dev.off()
```

pdf: 2

Raw output fo the mutation track from the above plot

Check in frame; change colour in plots

In [72]:
```
ord <- order(ex.by.mut[['mut']][,4], decreasing=TRUE)
head(ex.by.mut[['mut']][ord,])
```

	TRACK_ID	AJUBA_MUT	Alt.type	JUB.84962	IQR.90.
13	TCGA-CV-6003-01	T337_C341del	Nonsense	2.65706121	3
11	TCGA-CR-6493-01	C406S	Missense	0.72805934	2
25	TCGA-CX-7082-01	H360Y	Missense	0.69803062	1
6	TCGA-CN-6997-01	R50Efs*192	Nonsense	0.35677236	1
15	TCGA-CV-7099-01	H423Y	Missense	-0.06802904	0
10	TCGA-CR-6491-01	Q103*	Nonsense	-0.37074236	1

ADAM10

In [503]:
```
ex.by.mut <- parseIdsByMutation('ADAM10', getGeneExp('ADAM10', z.ex),
                                seg.ids=seg.ids, lo.q=0.1, hi.q=0.9,
                                cbio=cbio.abs, keep.only.abs=TRUE)
null <- .mutBoxplot(ex.by.mut, add.purity=FALSE)
lapply(ex.by.mut, head, 3)
```

$`NA`

TRACK_ID	Alt	Alt.type	ADAM10.102	IQR.90.
TCGA-BA-4074-01	no_alteration	NA	-0.8386675	1
TCGA-BA-4075-01	no_alteration	NA	-0.9267641	3
TCGA-BA-4076-01	no_alteration	NA	0.1363440	2

$mut

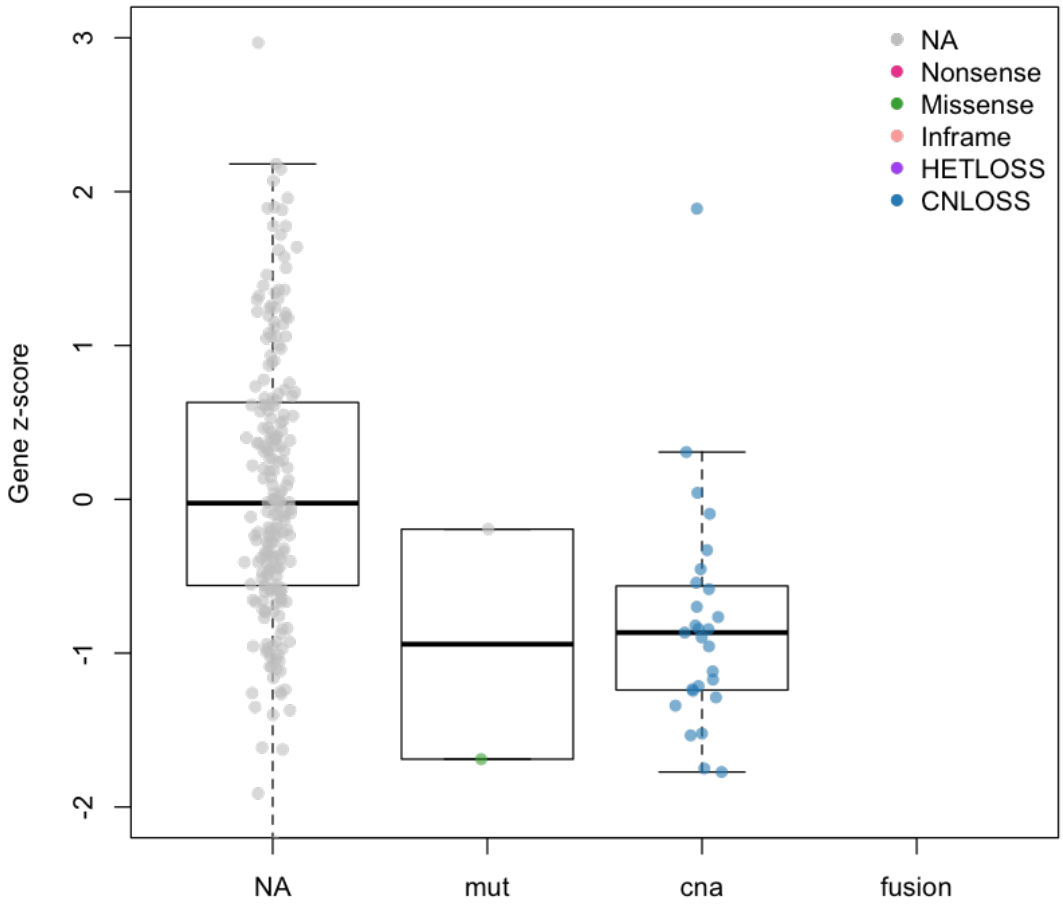
TRACK_ID	ADAM10_MUT	Alt.type	ADAM10.102	IQR.90.
TCGA-4P-AA8J-01	I440V	Missense	NA	1
TCGA-CV-7261-01	N278S,D121H	NA	-0.1942347	3
TCGA-IQ-7631-01	R74Q	Missense	-1.6890990	1

$cna

	TRACK_ID	ADAM10_CNA	Alt.type	ADAM10.102	IQR.90.
2	TCGA-BA-5153-01	CNLOSS	CNLOSS	-1.171536	1
3	TCGA-BA-5555-01	CNLOSS	CNLOSS	-0.765185	2
7	TCGA-BA-A4IF-01	CNLOSS	CNLOSS	NA	2

$fusion

TRACK_ID	ADAM10_FUSION	Alt.type	ADAM10.102	IQR.90.



In [508]:
```
t.test(ex.by.mut[['cna']][,4], ex.by.mut[['NA']][,4])
```

```
        Welch Two Sample t-test

data:  ex.by.mut[["cna"]][, 4] and ex.by.mut[["NA"]][, 4]
t = -5.7718, df = 37.653, p-value = 0.000001207
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.2269236 -0.5896055
sample estimates:
 mean of x  mean of y
-0.8018231  0.1064414
```

In [504]:
```
pdf(file.path(outdir, "adam10_mut_expr.pdf"), width=5)
null <- .mutBoxplot(ex.by.mut, add.purity=FALSE, jitter=0.1)
dev.off()
```

pdf: 2

Form the cBioPortal mutations csv

Take the mutation metadata, as well as the CNV's called from either ABSOLUTE or straight from the metadata sheet downloaded from cBioportal, and then generate the dataframe needed for online webtool visualization

In [434]:
```
genes <- unique(sort(gsub("_.*", "", colnames(mut.attr))))
genes <- genes[-grep("TRACK", genes)]

cbio.abs <- apply(mut.attr, 1, SchramekLOH::cbioConv, genes=genes,
                  use.absolute=use.absolute, sample.stdres=sample.stdres)
cbio.abs <- do.call("rbind", cbio.abs)

cbio.meta <- apply(mut.attr, 1, SchramekLOH::cbioConv, genes=genes,
                  use.absolute=FALSE)
cbio.meta <- do.call("rbind", cbio.meta)
```

In [31]:
```
head(cbio.abs[grep('AJUBA', cbio.abs$Gene),],)
```

	Sample	Gene	mut	mut.type
3	TCGA-CN-6011-01	AJUBA	HETLOSS	CNA
9	TCGA-CN-6018-01	AJUBA	C270Wfs*10	TRUNC
11	TCGA-CN-6019-01	AJUBA	HETLOSS	CNA
15	TCGA-CN-6020-01	AJUBA	HETLOSS	CNA
28	TCGA-CQ-5327-01	AJUBA	Q76*	TRUNC
33	TCGA-CQ-5332-01	AJUBA	HETLOSS	CNA

In [32]:
```
head(cbio.meta[grep('AJUBA', cbio.meta$Gene),])
```

	Sample	Gene	mut	mut.type
4	TCGA-CN-6011-01	AJUBA	HETLOSS	CNA
10	TCGA-CN-6018-01	AJUBA	C270Wfs*10	TRUNC
13	TCGA-CN-6019-01	AJUBA	HETLOSS	CNA
15	TCGA-CN-6020-01	AJUBA	HETLOSS	CNA
27	TCGA-CQ-5327-01	AJUBA	Q76*	TRUNC
33	TCGA-CQ-5332-01	AJUBA	HETLOSS	CNA

Listing all the samples that were classified as HETLOSS but are no longer

In [33]:
```
list(c('MIRES_p_TCGA_151_SNP_N_GenomeWideSNP_6_F06_831494', 'cnloh', 'AJUBA'),
          c('CLUBS_p_TCGA_186_188_SNP_N_GenomeWideSNP_6_E01_914086', 'purity', 'n
o_model'),
          c('PALPS_p_TCGA_265_266_267_N_GenomeWideSNP_6_A10_1306810', 'hetloss',
'no_model'),
          c('BUCKS_p_TCGA_272_273_N_GenomeWideSNP_6_C04_1320614', 'hetloss', 'no_
model'),
          c('CONGA_p_TCGA_b_317_318_319_NSP_GenomeWideSNP_6_F06_1365262', 'NEAT',
'AJUBA'),
          c('CONGA_p_TCGA_b_317_318_319_NSP_GenomeWideSNP_6_G11_1365318', 'hetlos
s'),
         c("MAULS_p_TCGA_189_190_SNP_N_GenomeWideSNP_6_C11_932832 ", "NEAT", "AJU
BA"))
```

1. 'MIRES_p_TCGA_151_SNP_N_GenomeWideSNP_6_F06_831494' 'cnloh' 'AJUBA'
2. 'CLUBS_p_TCGA_186_188_SNP_N_GenomeWideSNP_6_E01_914086' 'purity' 'no_model'
3. 'PALPS_p_TCGA_265_266_267_N_GenomeWideSNP_6_A10_1306810' 'hetloss' 'no_model'
4. 'BUCKS_p_TCGA_272_273_N_GenomeWideSNP_6_C04_1320614' 'hetloss' 'no_model'
5. 'CONGA_p_TCGA_b_317_318_319_NSP_GenomeWideSNP_6_F06_1365262' 'NEAT' 'AJUBA'
6. 'CONGA_p_TCGA_b_317_318_319_NSP_GenomeWideSNP_6_G11_1365318' 'hetloss'
7. 'MAULS_p_TCGA_189_190_SNP_N_GenomeWideSNP_6_C11_932832 ' 'NEAT' 'AJUBA'

In [34]:
```
hetloss.abs <- as.character(cbio.abs$Sample[which(with(cbio.abs, Gene == 'RIPK4' &
mut == 'HETLOSS'))])
hetloss.meta <- as.character(cbio.meta$Sample[which(with(cbio.meta, Gene == 'RIPK4
' & mut == 'HETLOSS'))])
if(0){
    data.frame("affy"=mapIds(setdiff(hetloss.meta, hetloss.abs), mapping.cov, in.t
ype='tcga', out.type='affy'),
               "tcga"=setdiff(hetloss.meta, hetloss.abs))
}
```

In [1633]:
```
## Isolate the samples containing a mutation on a particular gene
write.table(cbio.abs[which(cbio.abs$Gene == 'ADAM10'),],
           file=file.path(outdir, "adam10_absolute.tsv"),
           sep="\t", col.names=FALSE, quote=FALSE, row.names=FALSE)

write.table(cbio.abs[which(cbio.abs$Gene == 'AJUBA'),],
           file=file.path(outdir, "ajuba_absolute.tsv"),
           sep="\t", col.names=FALSE, quote=FALSE, row.names=FALSE)

write.table(cbio.abs[which(cbio.abs$Gene == 'RIPK4'),],
           file=file.path(outdir, "ripk4_absolute.tsv"),
           sep="\t", col.names=FALSE, quote=FALSE, row.names=FALSE)

## cBioportal TSV to generate the oncoprint
write.table(cbio.abs, file=file.path(outdir, "cbioportal.tsv"),
            sep="\t", col.names=FALSE, quote=FALSE, row.names=FALSE)
```

Final modification for cBioportal paper figure

You can use this Onco Query Language (OQL) to write the query:

```
NOTCH1: MUT HOMDEL
NOTCH2: MUT HOMDEL
NOTCH3: MUT HOMDEL
NOTCH4: MUT HOMDEL
JAG1: MUT HOMDEL
JAG2: MUT HOMDEL

ADAM10: MUT HOMDEL HETLOSS
AJUBA: MUT HOMDEL HETLOSS
RIPK4: MUT HOMDEL
```

In [450]:
```
.genBlankDf <- function(cbio){
    missing.samples <- mut.attr$TRACK_ID[which(! mut.attr$TRACK_ID %in% cbio$Sampl
e)]
    blank <- vector(mode = "character", length=length(missing.samples))
    blank <- rep("", length(missing.samples))
    blank.df <- data.frame("Sample"=missing.samples)
    blank.df
}
```

In [452]:
```
cbio.final <- cbio.abs
rm.idx <- lapply(c("NOTCH1", "NOTCH2", "NOTCH3", "NOTCH4", "JAG1", "JAG2", "RIPK4"
), function(gene){
    with(cbio.final, which(Gene == gene & mut != 'HOMDEL' & mut.type == 'CNA'))
})
rm.idx <- as.integer(as.character(unlist(rm.idx)))
cbio.final <- cbio.final[-rm.idx,]

#cbio.final <- rbind(cbio.abs, .genBlankDf(cbio.abs))
write.table(cbio.final, file=file.path(outdir, "cbioportal_final.tsv"),
            sep="\t", col.names=FALSE, quote=FALSE, row.names=FALSE)
write.table( .genBlankDf(cbio.final), file=file.path(outdir, "cbioportal_final.tsv
"),
            sep="\t", col.names=FALSE, quote=FALSE, row.names=FALSE, append=TRUE)
```

In [466]:
```
cnt <- sapply(c("ADAM10", "AJUBA", "RIPK4"), function(gene)
    nrow(cbio.final[with(cbio.final, which(Gene == gene & mut != 'HETLOSS')),]))
data.frame("A"=cnt, "x"=(cnt / 504), "y"=(cnt / 504) * 100)

cnt <- sapply(c("ADAM10", "AJUBA"), function(gene)
    nrow(cbio.final[with(cbio.final, which(Gene == gene & mut == 'HETLOSS')),]))
data.frame("A"=cnt, "x"=(cnt / 504), "y"=(cnt / 504) * 100)
```

	A	x	y
ADAM10	4	0.007936508	0.7936508
AJUBA	38	0.075396825	7.5396825
RIPK4	15	0.029761905	2.9761905

	A	x	y
ADAM10	64	0.1269841	12.69841
AJUBA	56	0.1111111	11.11111

Logistic regression predicting HETLOSS using StdRes and Purity

Combine the Seg_MAF from ABSOLUTE, with purity estimates, and the StdRes of LOH estimation into a single dataframe.

Then using just the HETLOSS samples and the Diploid samples, predict LOH based on just the StdRes of LOH with purity as a predictor. With this trained model, I can then use it to predict LHO in all other samples and genes. Additionally, I can combine multiple samples using the arithmetic mean of the probabilities for each bin to create gistic-like plots

In [188]:
```
head(sdf)
```

seqnames	start	end	width	strand	ID	sample
chr1	61735	564621	502887	*	TCGA-CN-6011-01	FISTS_p_TCGA_b107_121_SNP_N_Genome
chr1	603590	6461631	5858042	*	TCGA-CN-6011-01	FISTS_p_TCGA_b107_121_SNP_N_Genome
chr1	603590	6461631	5858042	*	TCGA-CN-6011-01	FISTS_p_TCGA_b107_121_SNP_N_Genome
chr1	603590	6461631	5858042	*	TCGA-CN-6011-01	FISTS_p_TCGA_b107_121_SNP_N_Genome
chr1	603590	6461631	5858042	*	TCGA-CN-6011-01	FISTS_p_TCGA_b107_121_SNP_N_Genome
chr1	603590	6461631	5858042	*	TCGA-CN-6011-01	FISTS_p_TCGA_b107_121_SNP_N_Genome

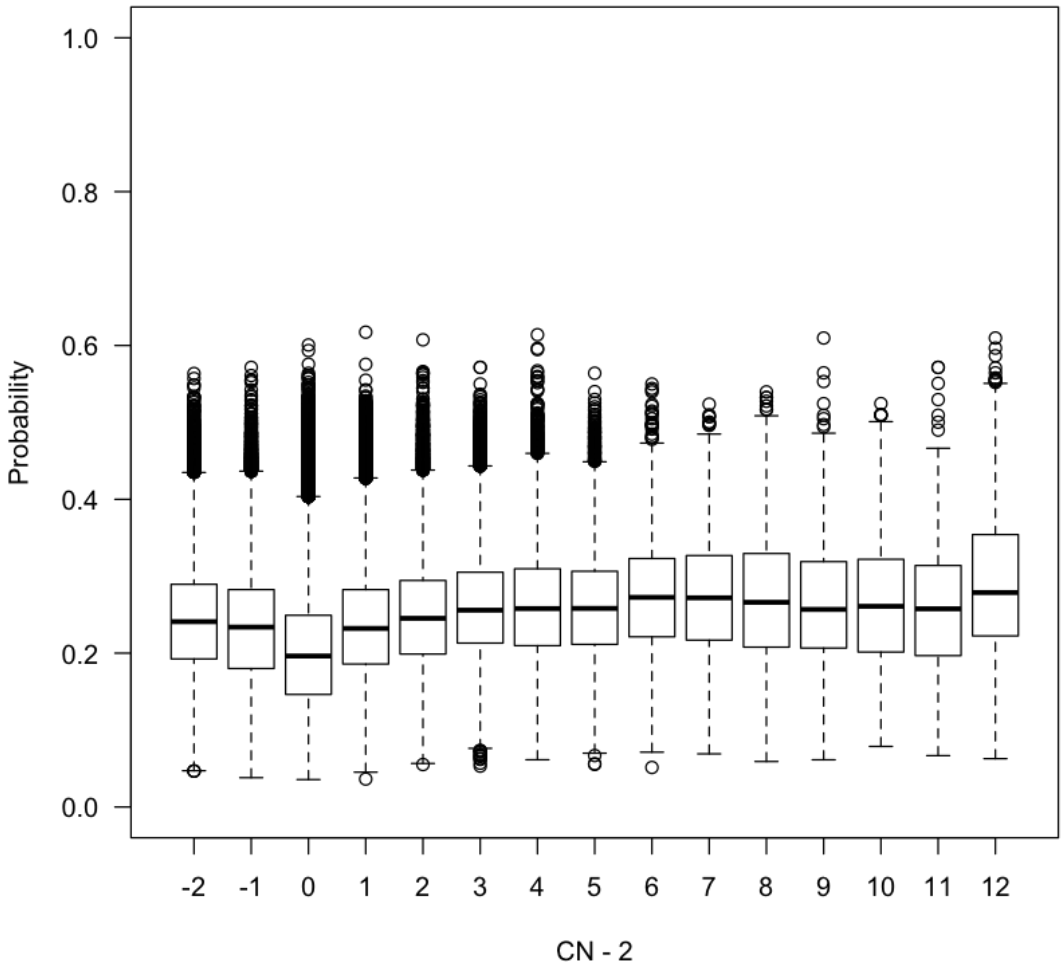
In [35]:
```
sdf <- lapply(names(all.stdres), combineSegLohPurity, seg.ids=seg.ids, all.stdres=
all.stdres)
```

In [36]:
```
sdf <- do.call("rbind", sdf)
sdf$HETLOSS <- FALSE
sdf[which(sdf$seg.mean == -1),]$HETLOSS <- TRUE
sdf$mean <- sdf$stdres
```

In [40]:
```
train.sdf <- sdf[which(sdf$seg.mean == -1 | sdf$seg.mean == 0), c('HETLOSS', 'copy
_ratio', 'mean',
                                                                  'purity', 'subcl
onal')]

glm.fit <- glm(HETLOSS ~ mean + purity, data = train.sdf, family = binomial)
summary(glm.fit)
glm.probs <- predict(glm.fit,type = "response")
```

```
Call:
glm(formula = HETLOSS ~ mean + purity, family = binomial, data = train.sdf)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3549  -0.7301  -0.6107  -0.4626   2.5562

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.464736   0.007716  -60.23   <2e-16 ***
mean        -0.081027   0.000701 -115.59   <2e-16 ***
purity      -2.284907   0.015343 -148.92   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1130841  on 1103240  degrees of freedom
Residual deviance: 1095929  on 1103238  degrees of freedom
  (28416 observations deleted due to missingness)
AIC: 1095935

Number of Fisher Scoring iterations: 4
```

In [1958]:
```
boxplot(lapply(split(sdf, f=sdf$seg.mean),
               function(x) predict(glm.fit, x, type='response')),
                   ylab="Probability", xlab="CN - 2", las=1,
                   ylim=c(0,1))
```



Some formatting of the combined seg + stdres + purity data structure to split it based on samples, and then calculate the LOH probability using the trained logistic model

In [41]:
```
## Create a LOH/Allelic imbalance probability matrix
sdf.spl <- split(sdf[,-c(1:5)], f=sdf$ID)
loh.list <- sapply(sdf.spl, getLohProb, mod=glm.fit)
loh.mat <- round(do.call("cbind", loh.list), 2)

## Format the position of each segment
loh.pos <- sdf.spl[[1]][,c("seqnames", "start", "end")]
loh.pos <- loh.pos[-which(duplicated(loh.pos)), ]
```

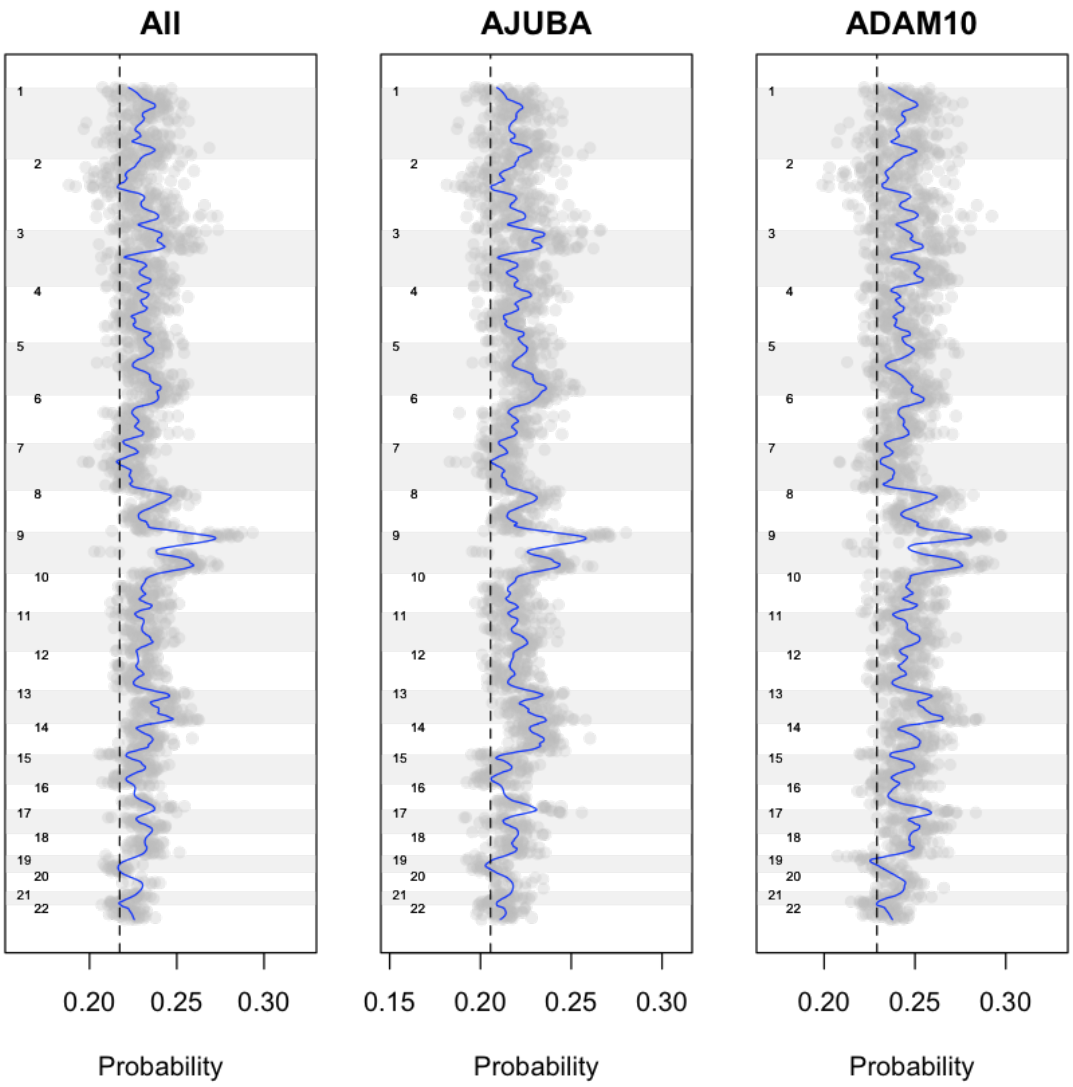
```
Warning message in cbind(...):
“number of rows of result is not a multiple of vector length (arg 2)”
```

Visualization of LOH Prob

In [243]:
```
#pdf(file.path(outdir, "loh_plots.pdf"))
split.screen(c(1,3))
screen(1); par(mar=c(5.1, 1, 2, 1)); all.loess <- plotLohProb(main="All")
screen(2); par(mar=c(5.1, 1, 2, 1)); ajuba.loess <- plotLohProb("AJUBA")
screen(3); par(mar=c(5.1, 1, 2, 1)); adam.loess <- plotLohProb("ADAM10")
close.screen(all.screens=TRUE)
#dev.off()
```

```
1 2 3
```

```
[1] "Processing LOH..."
[1] "Processing LOH..."
[1] "Processing LOH..."
```

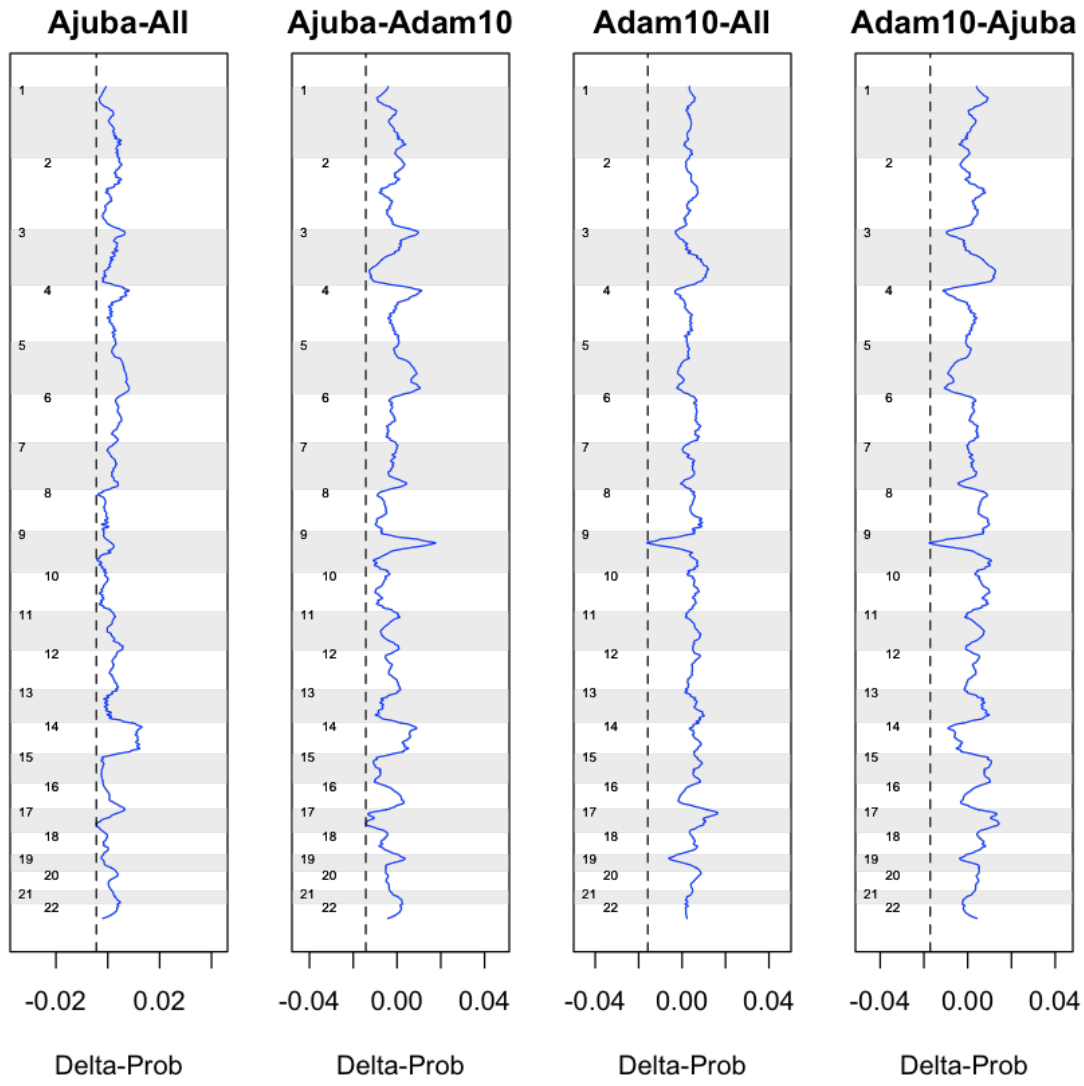


In [269]:
```
plotLohProb <- function(goi=NULL, soi=NULL, i=NULL, j=NULL, diff=FALSE,
                        main=NULL, is.expr=FALSE, xlab='Probability',
                        minx=NULL){
  if(is.null(goi) & is.null(soi)){
    soi <- as.character(cbio.abs$Sample)
    soi <- soi[soi %in% colnames(loh.mat)]
    goi <- 'all'
  } else if (is.null(soi)) {
    soi <- as.character(cbio.abs[which(cbio.abs$Gene == goi),]$Sample)
    main <- goi
  }

  if(!is.expr){
    print("Processing LOH...")
    loh <- data.frame("prob"=apply(loh.mat[,soi], 1, mean),
                      "idx"=c(1:nrow(loh.mat)))
    rle.x <- rle(as.character(loh.pos$seqnames))
    na.chr <- which(!is.na(rle.x$values))
    chr.xpos <- c(1, cumsum(rle.x$length))
  } else {
    print("Processing expression...")
    soi <- gsub("-", ".", soi)

    gaf.spl <- split(gaf.ord, f=gaf.ord$chr)
    cum.x <- sapply(gaf.spl, function(x) c(min(x$start), diff(x$start)))
    cum.x <- cumsum(as.numeric(unlist(cum.x[paste0("chr", c(1:22, "X", "Y"))])))
    length(cum.x) <- nrow(gaf.ord)
    gaf.ord$cumx <- cum.x

    loh <- data.frame("prob"=apply(z.ex[,soi], 1, function(x) mean(as.numeric(x),
na.rm=TRUE)),
                      "idx"=gaf.ord$cumx)

    rle.x <- rle(as.character(gaf.ord$chr))
    na.chr <- which(!is.na(rle.x$values))
    chg <- c(1, cumsum(rle.x$length))
    chr.xpos <- na.omit(gaf.ord$cumx[chg])
  }



  # Fit the Loess Model
  loessMod10 <- loess(prob ~ idx, data=loh, span=0.03) # 10% smoothing span
  smoothed10 <- predict(loessMod10, loh$idx)

  # Plot the loess model
  if(diff){
    smoothed10 <- (i - min(i, na.rm=TRUE)) - (j - min(j, na.rm=TRUE))
    min.x <- min(smoothed10, na.rm=TRUE) - 0.03
    max.x <- max(smoothed10, na.rm=TRUE) + 0.03
  } else {
    min.x <- min(loh$prob, na.rm=TRUE) - 0.03
    max.x <- max(loh$prob, na.rm=TRUE) + 0.03
  }

  with(loh, plot((prob), -1*idx, pch=16, col=alpha("grey", 0.3),
                 xlim=if(!is.null(minx)) minx else c(min.x, max.x), las=1,
                 main=main, yaxt='n', ylab='', xlab=xlab,
                 type=if(diff) 'n' else 'p'))
  rect(ytop=-1*chr.xpos[-length(chr.xpos)][c(TRUE,FALSE)],  xleft = -10,
       ybottom = -1*chr.xpos[-1][c(TRUE,FALSE)], xright = 10,
       border=FALSE, col=alpha("grey", 0.3))
  lines(smoothed10, -1*(loh$idx), col="blue")
```

In [272]:
```
split.screen(c(1,4))
screen(1); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=ajuba.loess, j=all.loes
s, diff=TRUE, main="Ajuba-All", xlab='Delta-Prob')
screen(2); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=ajuba.loess, j=adam.loe
ss, diff=TRUE, main="Ajuba-Adam10", xlab='Delta-Prob')
screen(3); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=adam.loess, j=all.loess
, diff=TRUE, main="Adam10-All", xlab='Delta-Prob')
screen(4); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=adam.loess, j=ajuba.loe
ss, diff=TRUE, main="Adam10-Ajuba", xlab='Delta-Prob')
close.screen(all.screens=TRUE)
```

```
 1  2  3  4
```

```
[1] "Processing LOH..."
[1] "Processing LOH..."
[1] "Processing LOH..."
[1] "Processing LOH..."
```



Output for Paper figures

In [274]:
```
pdf(file.path(outdir, "gistic_plots", "delta_loess.pdf"))
split.screen(c(1,4))
screen(1); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=ajuba.loess, j=all.loes
s, diff=TRUE, main="Ajuba-All", xlab='Delta-Prob')
screen(2); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=ajuba.loess, j=adam.loe
ss, diff=TRUE, main="Ajuba-Adam10", xlab='Delta-Prob')
screen(3); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=adam.loess, j=all.loess
, diff=TRUE, main="Adam10-All", xlab='Delta-Prob')
screen(4); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(i=adam.loess, j=ajuba.loe
ss, diff=TRUE, main="Adam10-Ajuba", xlab='Delta-Prob')
close.screen(all.screens=TRUE)
dev.off()
```

```
    1  2  3  4
```

```
[1] "Processing LOH..."
[1] "Processing LOH..."
[1] "Processing LOH..."
[1] "Processing LOH..."
```

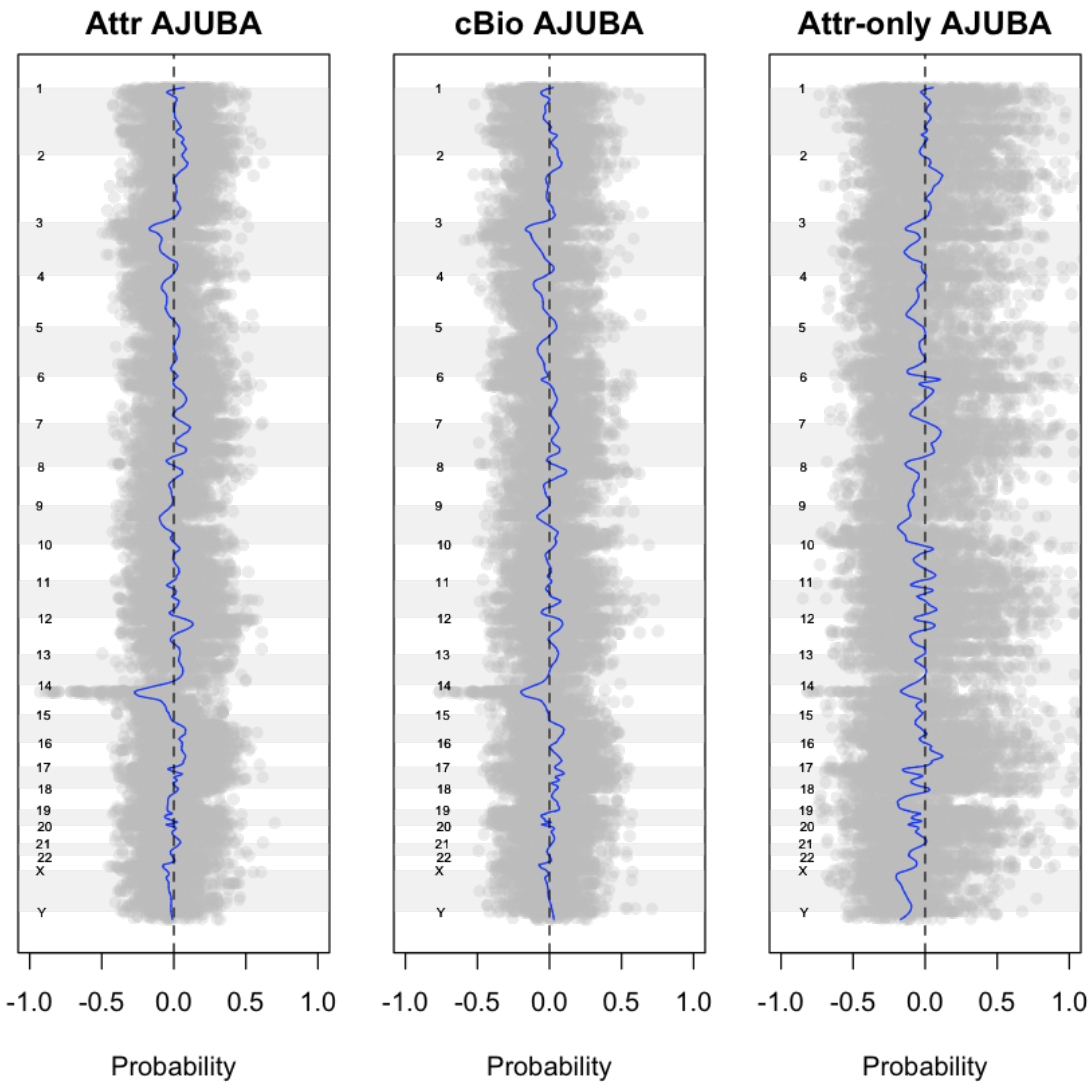
pdf: 2

Visualizing summary of gene z-scores

The point of this is to visualize the expression z-scores of samples containing mutations in certain genes

In [245]:
```
adam10.idx <- which(mut.attr$AJUBA_CNA != 'no_alteration')
attr.soi <- mut.attr[adam10.idx, ]$TRACK_ID
cbio.soi <- as.character(cbio.abs[which(cbio.abs$Gene == 'AJUBA'),]$Sample)

#pdf(file.path(outdir, "expr_plots.pdf"))
split.screen(c(1,3))
screen(1); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(soi=attr.soi, is.expr=TRU
E,
                                                         main='Attr AJUBA', minx=c
(-1,1))
screen(2); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(soi=cbio.soi, is.expr=TRU
E,
                                                         main='cBio AJUBA', minx=c
(-1,1))
screen(3); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(soi=setdiff(attr.soi, cbi
o.soi), is.expr=TRUE,
                                                         main='Attr-only AJUBA', m
inx=c(-1,1))
close.screen(all.screens=TRUE)
#dev.off()
```

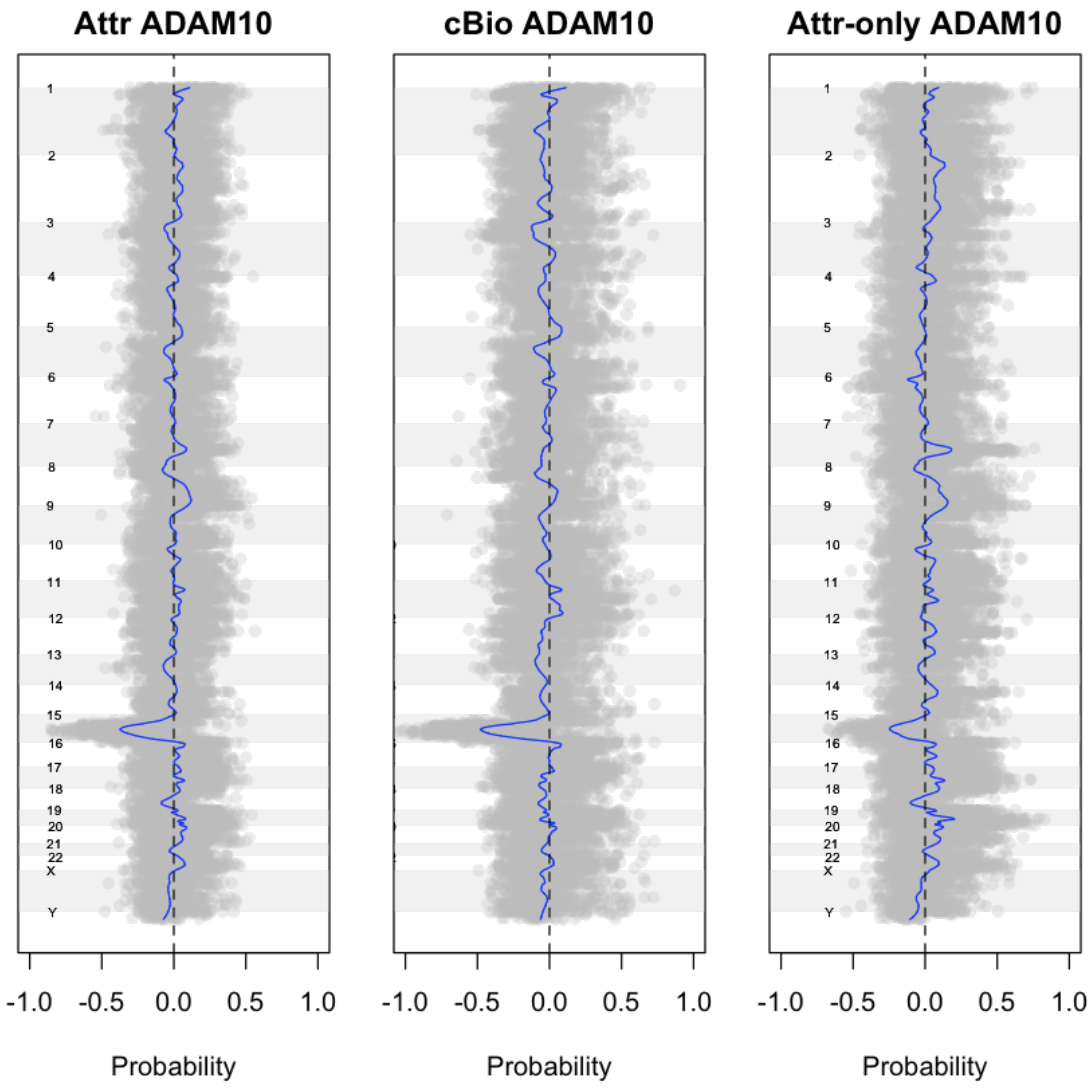
```
  1  2  3

[1] "Processing expression..."
[1] "Processing expression..."
[1] "Processing expression..."
```



In [246]:
```
adam10.idx <- which(mut.attr$ADAM10_CNA != 'no_alteration')
attr.soi <- mut.attr[adam10.idx, ]$TRACK_ID
cbio.soi <- as.character(cbio.abs[which(cbio.abs$Gene == 'ADAM10'),]$Sample)

#pdf(file.path(outdir, "expr_plots.pdf"))
split.screen(c(1,3))
screen(1); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(soi=attr.soi, is.expr=TRUE,
                                                         main='Attr ADAM10', minx=c(-1,1))
screen(2); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(soi=cbio.soi, is.expr=TRUE,
                                                         main='cBio ADAM10', minx=c(-1,1))
screen(3); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(soi=setdiff(attr.soi, cbio.soi), is.expr=TRUE,
                                                         main='Attr-only ADAM10', minx=c(-1,1))
close.screen(all.screens=TRUE)
#dev.off()
```

```
   1  2  3
```

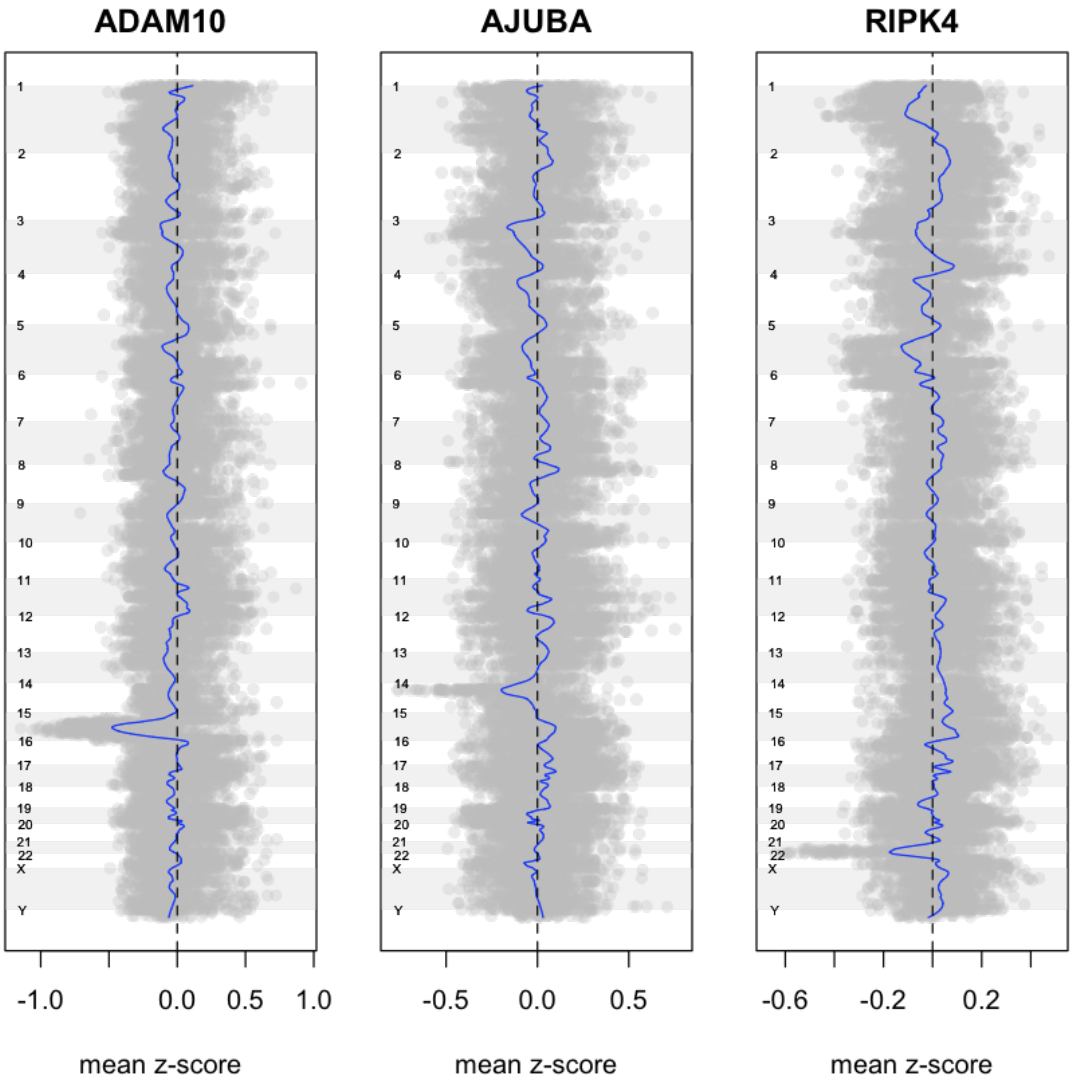
```
[1] "Processing expression..."
[1] "Processing expression..."
[1] "Processing expression..."
```



In [249]:
```
#pdf(file.path(outdir, "expr_plots.pdf"))
split.screen(c(1,3))
screen(1); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='ADAM10', is.expr=TRU
E, xlab='mean z-score')
screen(2); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='AJUBA', is.expr=TRUE
, xlab='mean z-score')
screen(3); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='RIPK4', is.expr=TRUE
, xlab='mean z-score')
close.screen(all.screens=TRUE)
#dev.off()
```

```
  1  2  3
```

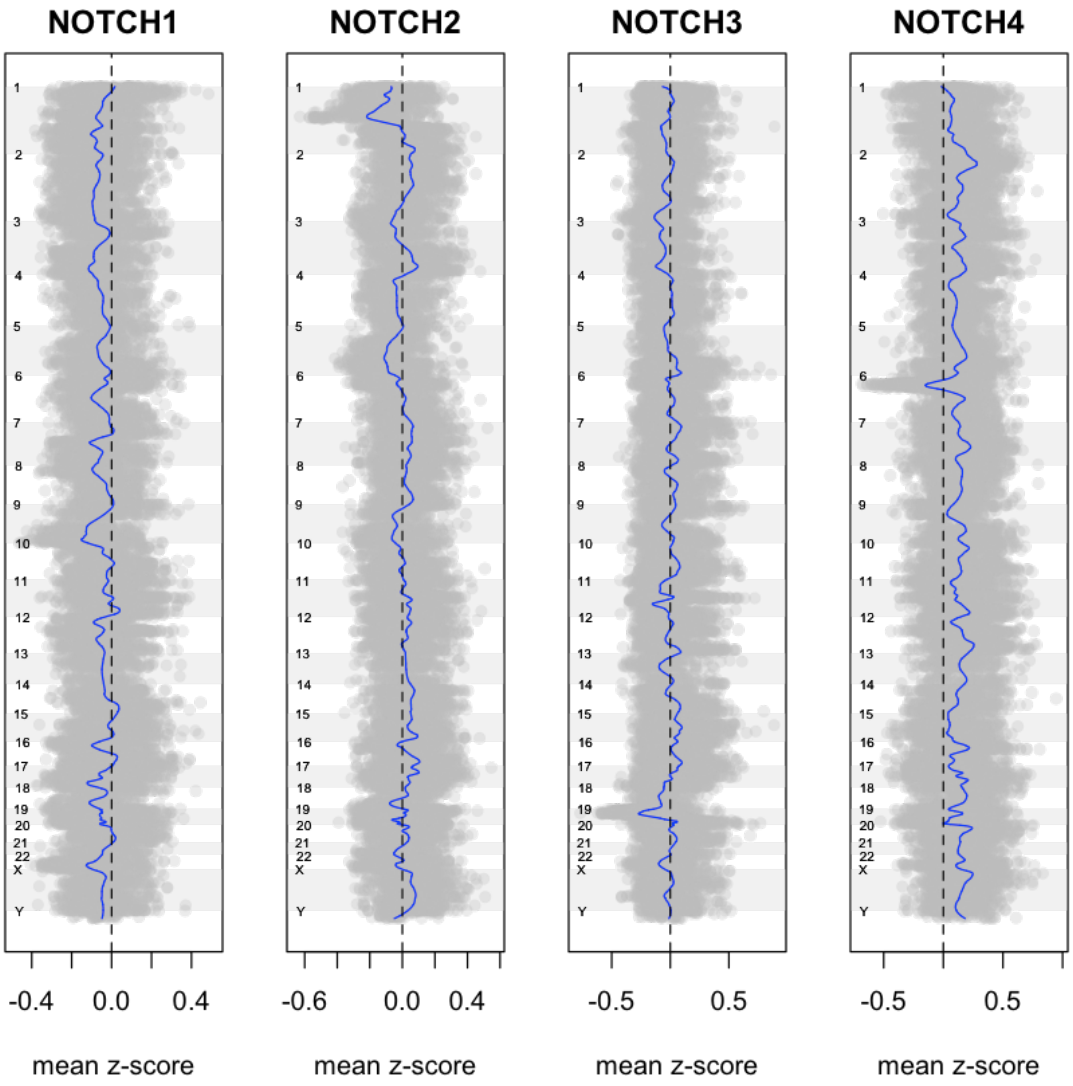
```
[1] "Processing expression..."
[1] "Processing expression..."
[1] "Processing expression..."
```



In [248]:
```
split.screen(c(1,4))
screen(1); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='NOTCH1', is.expr=TRUE, xlab='mean z-score')
screen(2); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='NOTCH2', is.expr=TRUE, xlab='mean z-score')
screen(3); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='NOTCH3', is.expr=TRUE, xlab='mean z-score')
screen(4); par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi='NOTCH4', is.expr=TRUE, xlab='mean z-score')
close.screen(all.screens=TRUE)
```

```
    1  2  3  4
```

```
[1] "Processing expression..."
[1] "Processing expression..."
[1] "Processing expression..."
[1] "Processing expression..."
```



Generating figures for the paper

In [268]:
```
dir.create(file.path(outdir, "gistic_plots"), recursive = TRUE, showWarnings = FAL
SE)

.outExpr <- function(gene){
    png(file.path(outdir, "gistic_plots", paste0(gene, "_expr_plots.png")), width=
2.5, height=7.5, units = "in", res=300)
    par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi=gene, is.expr=TRUE, xlab='Me
an z-score')
    dev.off()
}
.outLoh <- function(gene){
    png(file.path(outdir, "gistic_plots", paste0(gene, "_loh_plots.png")), width=2
.5, height=7.5, units = "in", res=300)
    par(mar=c(5.1, 1, 2, 1)); null <- plotLohProb(goi=gene, xlab='Mean probability
')
    dev.off()
}

.outExpr("ADAM10")
.outLoh("ADAM10")
.outExpr("AJUBA")
.outLoh("AJUBA")
.outExpr("RIPK4")
.outLoh("RIPK4")
```

```
[1] "Processing expression..."
```

pdf: 2

```
[1] "Processing LOH..."
```

pdf: 2

```
[1] "Processing expression..."
```

pdf: 2

```
[1] "Processing LOH..."
```

pdf: 2

```
[1] "Processing expression..."
```

pdf: 2

```
[1] "Processing LOH..."
```

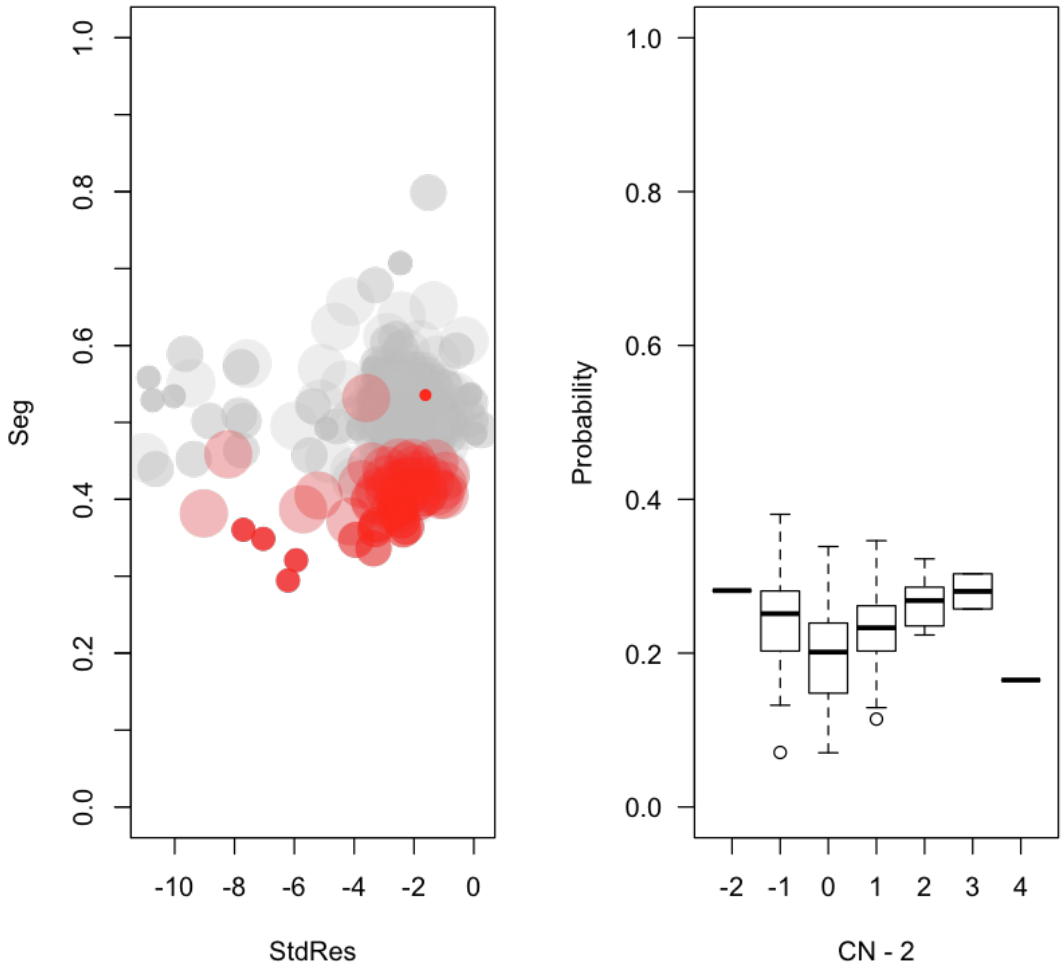
pdf: 2

Association between LRR and StdRes of HETLOSS Regions

In [1947]:
```
train.gene <- FALSE
```

In [1948]:
```
gene <- 'ADAM10'
mut <- paste0(gene, "_CNA")
rge <- 5
split.screen(c(1,2));
screen(1); range.stdres <- visOneGene(gene, rge, mut, use.absolute=TRUE,
                                      purity=purity, plot.cr=TRUE, use.seg=TRUE)
screen(2); boxplot(lapply(split(range.stdres, f=range.stdres$cn),
                  function(x) predict(glm.fit, x, type='response')),
                      ylab="Probability", xlab="CN - 2", las=1,
                      ylim=c(0,1))
close.screen(all.screens=TRUE)
```

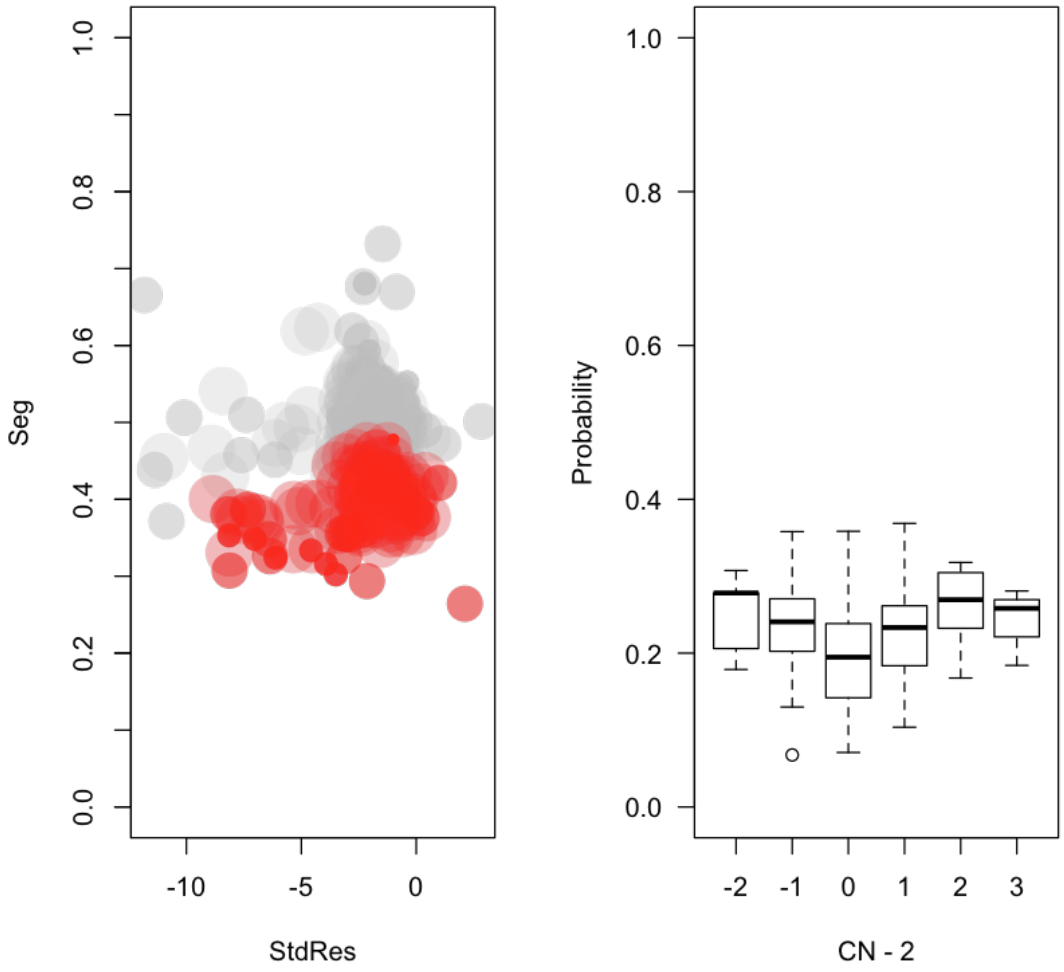
```
1  2
```



In [1742]:
```
if(train.gene){
    train <- range.stdres[which(range.stdres$cn == -1 | range.stdres$cn == 0), ]
    glm.fit <- glm(HETLOSS ~ mean  + purity, data = train, family = binomial)
    summary(glm.fit)
    glm.probs <- predict(glm.fit,type = "response")
}
```

In [1744]:
```
gene <- 'RIPK4'
mut <- paste0(gene, "_CNA")
rge <- 5
split.screen(c(1,2));
screen(1); range.stdres <- visOneGene(gene, rge, mut, use.absolute=TRUE,
                                      purity=purity, plot.cr=TRUE, use.seg=TRUE)
screen(2); boxplot(lapply(split(range.stdres, f=range.stdres$cn),
                  function(x) predict(glm.fit, x, type='response')),
                      ylab="Probability", xlab="CN - 2", las=1,
                      ylim=c(0,1))
close.screen(all.screens=TRUE)
```

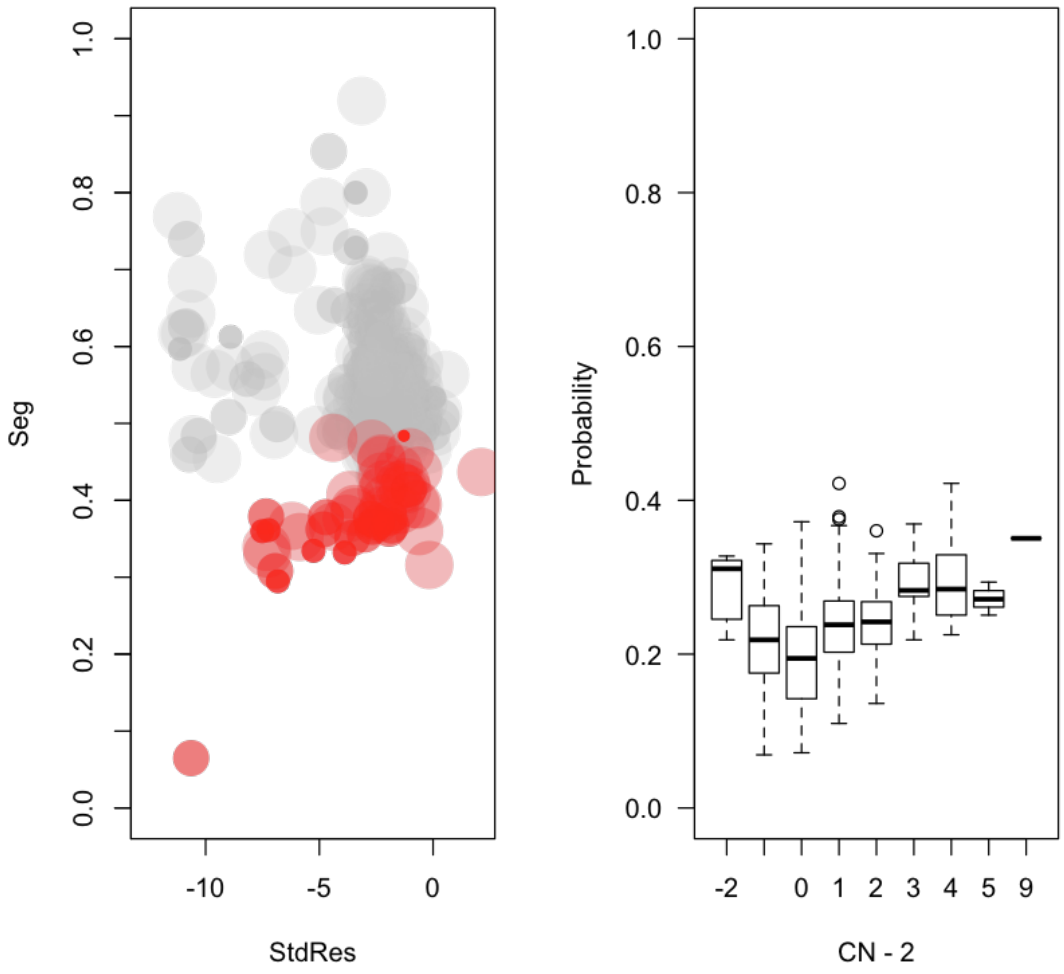
```
1  2
```



In [1747]:
```
if(train.gene){
    glm.fit <- glm(HETLOSS ~ mean  + purity, data = range.stdres, family = binomi
al)
    summary(glm.fit)
    glm.probs <- predict(glm.fit,type = "response")
}
```

In [1748]:
```
gene <- 'AJUBA'
mut <- paste0(gene, "_CNA")
rge <- 5

split.screen(c(1,2));
screen(1); range.stdres <- visOneGene(gene, rge, mut, use.absolute=TRUE,
                                      purity=purity, plot.cr=TRUE, use.seg=TRUE)
screen(2); boxplot(lapply(split(range.stdres, f=range.stdres$cn),
                  function(x) predict(glm.fit, x, type='response')),
                      ylab="Probability", xlab="CN - 2", las=1,
                      ylim=c(0,1))
close.screen(all.screens=TRUE)
```

```
1 2
```



In [1749]:
```
if(train.gene){
    glm.fit <- glm(HETLOSS ~ mean  + purity, data = range.stdres, family = binomi
al)
    summary(glm.fit)
    glm.probs <- predict(glm.fit,type = "response")
}
```

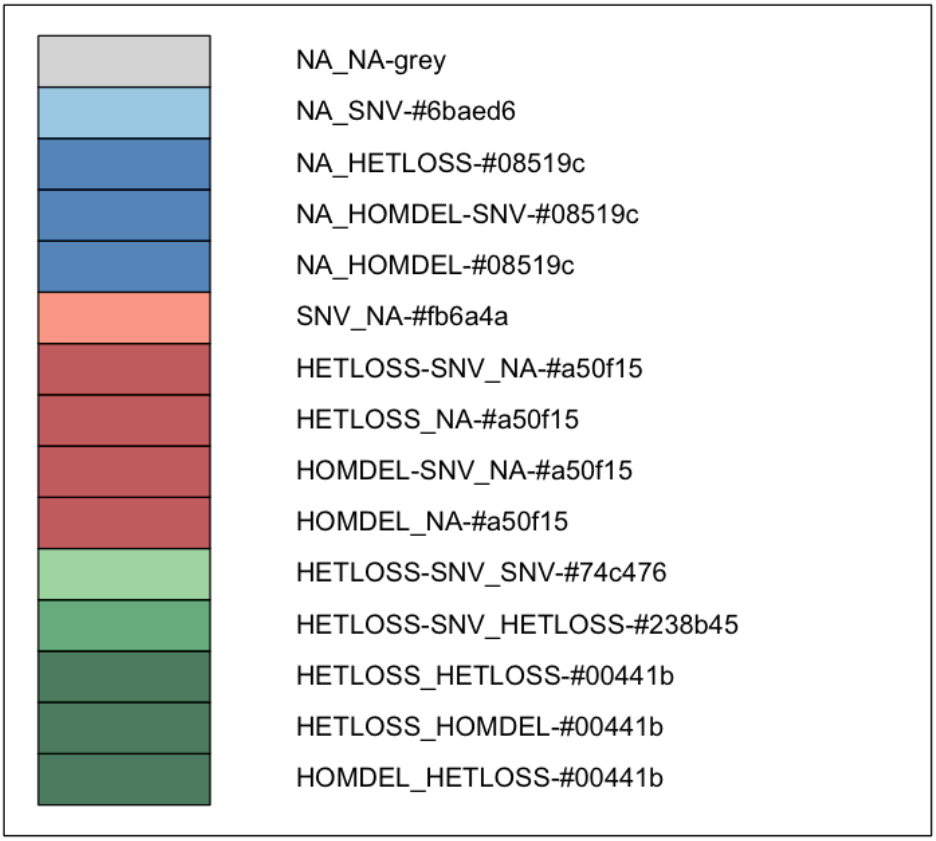
For interest, there was one AJUBA case that had Homozygous Deletion, and one AJUBA case where the logRRatio was EXTREMELY low. I wanted to see the details of these cases

Comparison of Seg/LOH between two genes

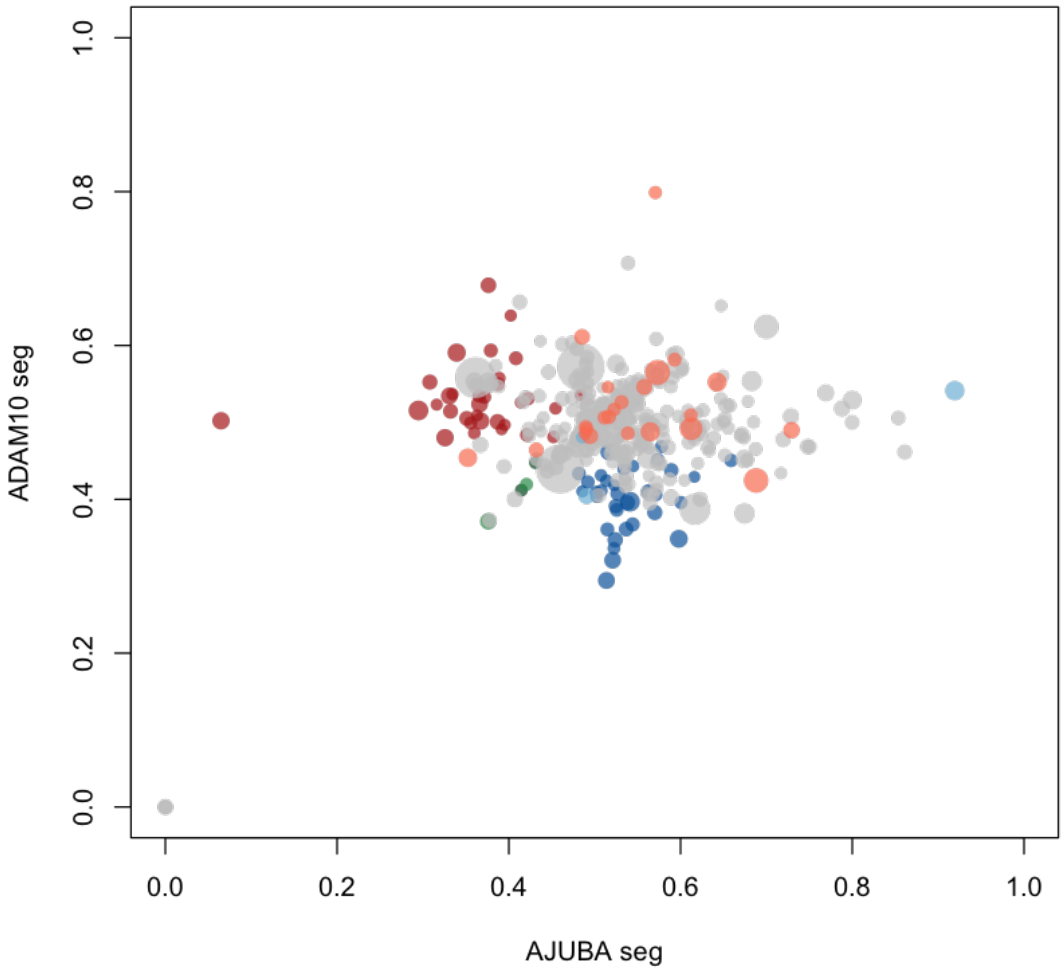
Create a colour schema for the unique ids (UID)

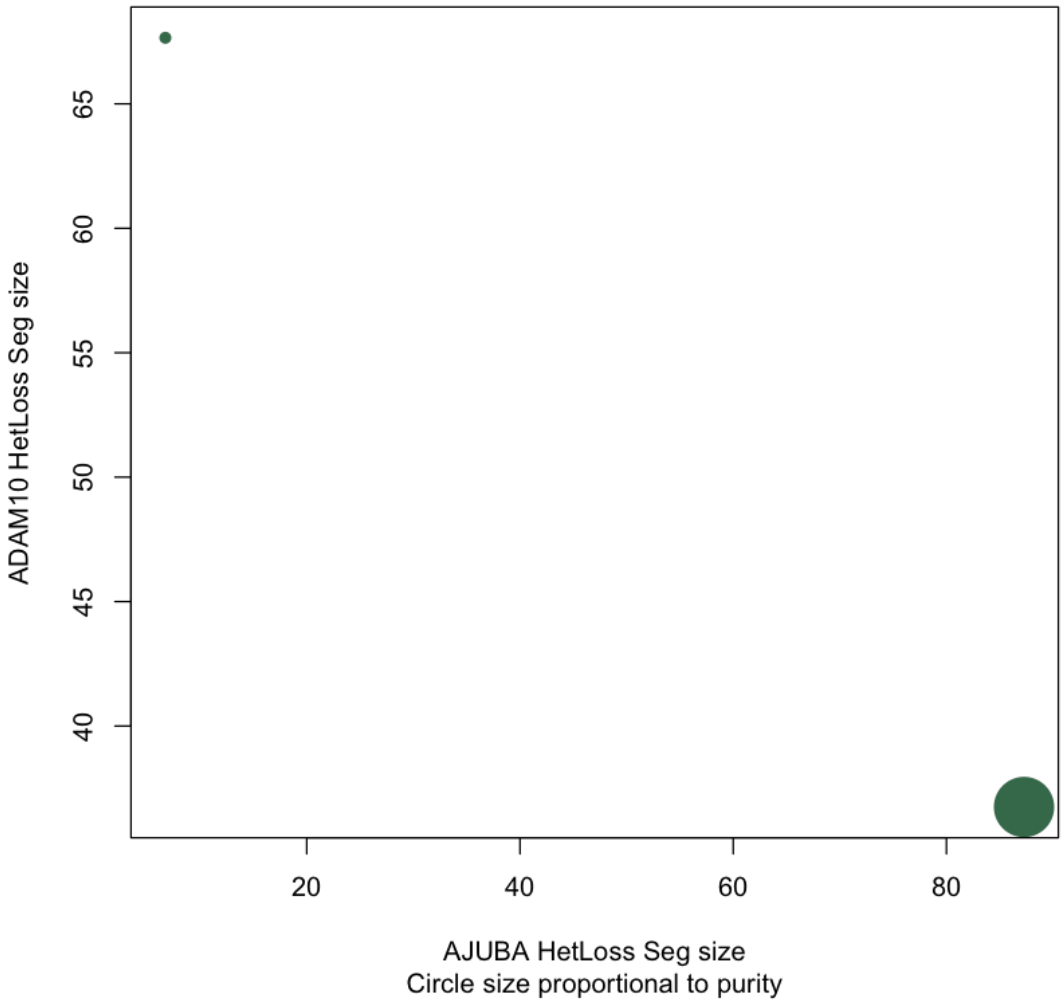
In [1455]:
```
col.df <- data.frame("UID"=c("HOMDEL_HETLOSS", "HETLOSS_HOMDEL", "HETLOSS_HETLOSS
", "HETLOSS-SNV_HETLOSS", "HETLOSS-SNV_SNV",
                                "HOMDEL_NA", "HOMDEL-SNV_NA", "HETLOSS_NA", "HETLOSS
-SNV_NA", "SNV_NA",
                                "NA_HOMDEL", "NA_HOMDEL-SNV", "NA_HETLOSS", "NA_SNV"
, "NA_NA"),

                     "col"=c("#00441b", "#00441b", "#00441b", "#238b45", "#74c476
",
                             "#a50f15", "#a50f15", "#a50f15", "#a50f15", "#fb6a4a"
,
                             "#08519c", "#08519c", "#08519c", "#6baed6", "grey"))
```

In [1555]:
```
gene.y <- 'ADAM10'
gene.x <- 'AJUBA'
rge <- 5

range.stdres.xyz <- compTwoGenes(gene.x, gene.y, col.df, cex.type='xy', use.absol
ute=use.absolute)
range.stdres.xyz[which(range.stdres.xyz$UID == 'HETLOSS_HETLOSS'),]
plotSegSizes()
```



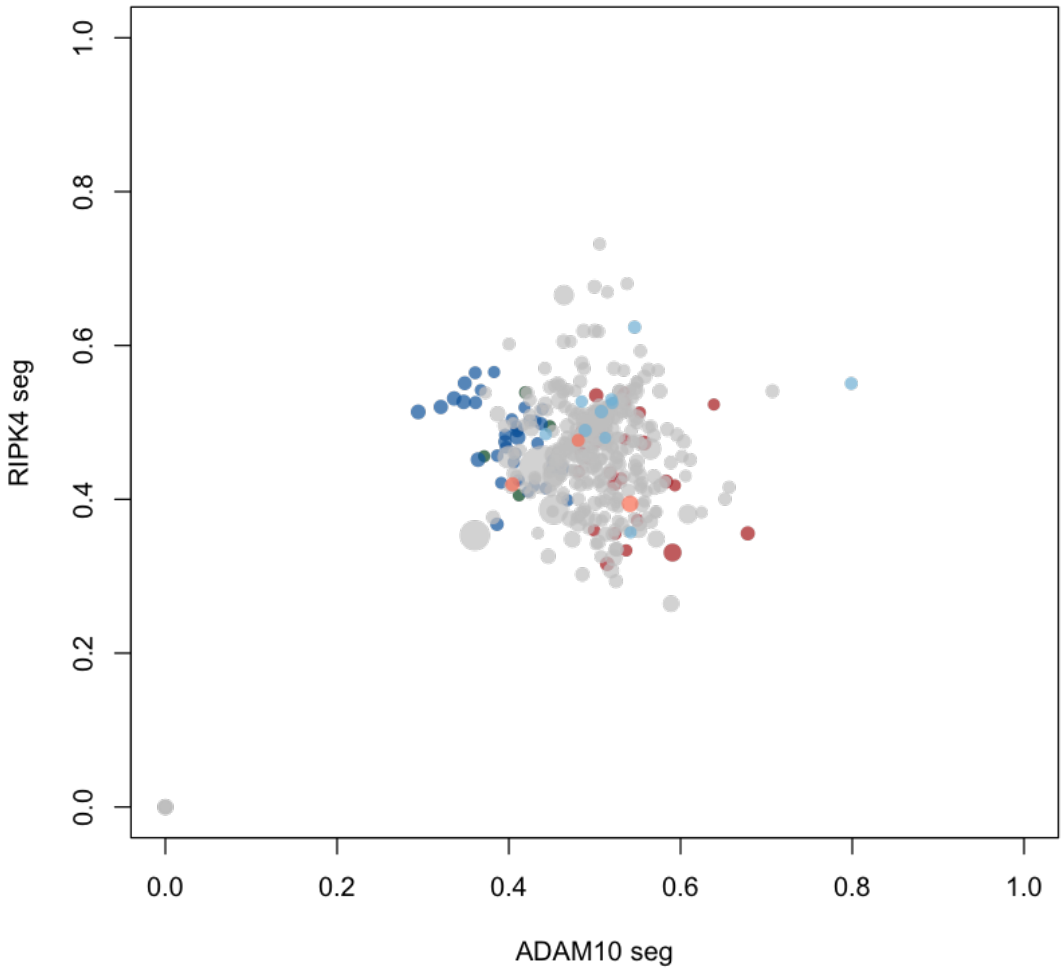
UID	TCGA_ID	TRACK_ID
HETLOSS_HETLOSS	TCGA-H7-A76A-01	UNDID_p_TCGA_353_354_355_37_NSP_GenomeWideSNP_6_G09_137
HETLOSS_HETLOSS	TCGA-CQ-7072-01	RICES_p_TCGA_Batch_310_311_NSP_GenomeWideSNP_6_D03_13618


ADAM10 seg
AJUBA seg
0.0
0.2
0.4
0.6
0.8
1.0


ADAM10 HetLoss Seg size
65
60
55
50
45
40
20
40
60
80
AJUBA HetLoss Seg size
Circle size proportional to purity

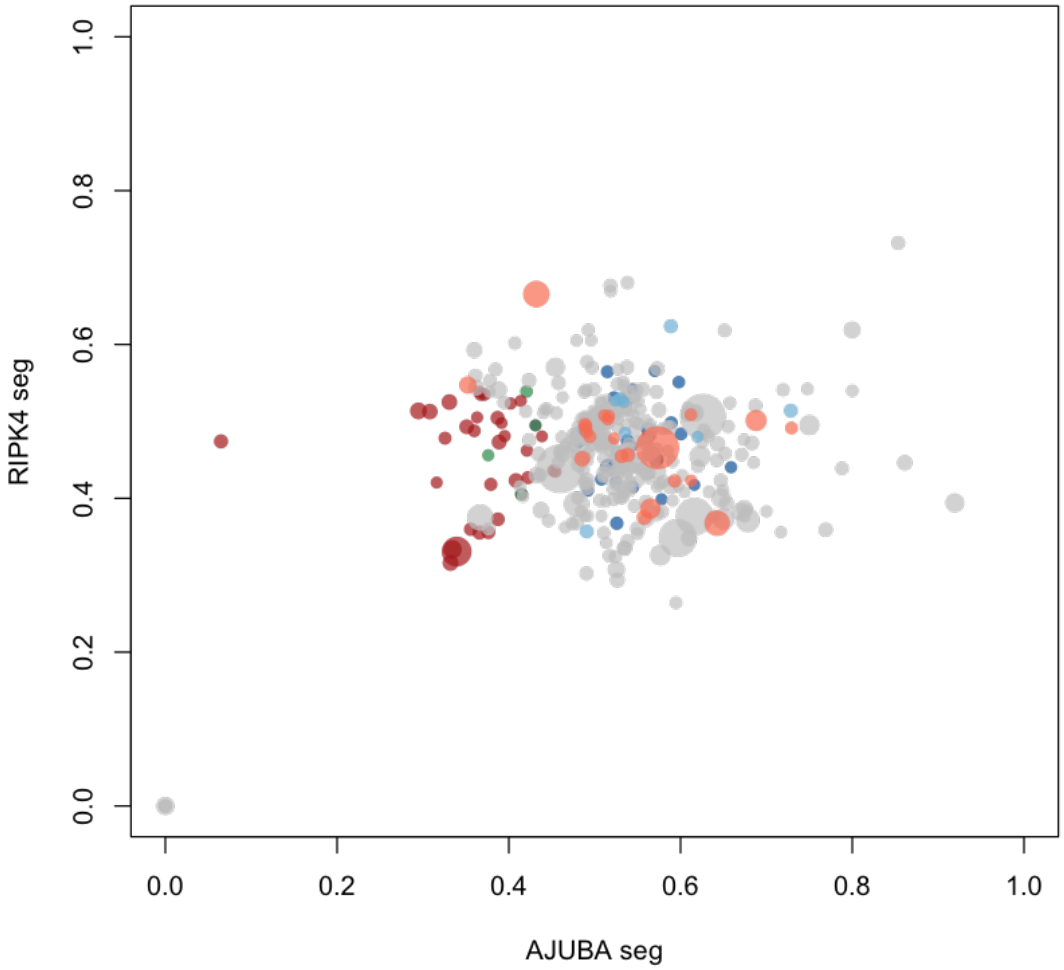
In [1621]:
```
range.stdres.xyz <- compTwoGenes('ADAM10', 'RIPK4', col.df, cex.type = 'xy',
                                 plot.legend=FALSE, use.absolute=use.absolute)
head(range.stdres.xyz[which(range.stdres.xyz$seg.y < -0.3),], 10)
#range.stdres.xyz[which(range.stdres.xyz$UID == 'HETLOSS_HETLOSS'),]
```

UID	TCGA_ID	TRACK_ID	mean.x	sd.x	seg.x	copy_ratio.x	seg.start.x	seg.end.x	unity.x	mean.y	s



In [1620]:
```
range.stdres.xyz <- compTwoGenes('AJUBA', 'RIPK4', col.df, cex.type = 'xy',
                                 use.absolute=use.absolute, plot.legend=FALSE)
head(range.stdres.xyz[which(range.stdres.xyz$seg.y < -0.3),], 10)
```

UID	TCGA_ID	TRACK_ID	mean.x	sd.x	seg.x	copy_ratio.x	seg.start.x	seg.end.x	unity.x	mean.y	s



In [ ]:
```
```