Data mining and genomic plots

Di Cook & Hadley Wickham
@visnut & @hadleywickham

July 2015

Classification

- Finding how two or more labelled classes differ on gene expression

- Typically multiple patients in each class

- Zeenia Jagga and Dinesh Gupta “Classification models for clear cell renal carcinoma stage progression, based on tumor RNAseq expression trained supervised machine learning algorithms” BMC Proceedings 2014, **8**(Suppl 6):S2

Example data

- 475 patients, diagnosed with early or late stage renal cancer
- 20,534 genes
- Patients split into 80% training and 20% test sets
- Genes filtered using WEKA down to just 62

Setting up

```
> tr <- read.csv("1753-6561-8-s6-s2-s2.csv")
> dim(tr)
[1]   380 20534
> gn <- colnames(tr[-c(1,20534)])
> sg <- scan("selected-genes.txt", what=character())
Read 62 items
> sga <- paste(sg, collapse="|")
> keep <- grepl(sga, gn)
> keep <- c(TRUE, keep, TRUE)
> tr.s <- tr[,keep]
> dim(tr.s)
[1] 380  68
> colnames(tr.s)
 [1] "Patient_id" “ALX1.8092" "AP1M1.8907" "APOL1.8542"
 [5] “C11orf73.51501" “C13orf16.121793" “C5orf62.85027"
> tr.s <- tr.s[,-c(13,19,32,39)]
> summary(tr.s)
             Patient_id    ALX1.8092          AP1M1.8907
  TCGA-A3-3306-01A:  1   Min.   :0.00000   Min.   : 1.54
  TCGA-A3-3308-01A:  1   1st Qu.:0.01289   1st Qu.:17.30
  TCGA-A3-3311-01A:  1   Median :0.04245   Median :20.25
  TCGA-A3-3313-01A:  1   Mean   :0.15402   Mean   :21.06
```

Transform

- Counts are very skewed, to get a reasonable classification model fit, need to transform by logs

- Should we do anything about different ranges, too?

```
> library(GGally)
> library(dplyr)
> tr.s[,2:63] <- log(tr.s[,2:63]+1)
> summary(tr.s)
             Patient_id    ALX1.8092          AP1M1.8907
  TCGA-A3-3306-01A:  1   Min.   :0.00000   Min.   :0.9321
  TCGA-A3-3308-01A:  1   1st Qu.:0.01281   1st Qu.:2.9071
  TCGA-A3-3311-01A:  1   Median :0.04157   Median :3.0564
  TCGA-A3-3313-01A:  1   Mean   :0.11719   Mean   :3.0549
  TCGA-A3-3316-01A:  1   3rd Qu.:0.14485   3rd Qu.:3.1989
  TCGA-A3-3317-01A:  1   Max.   :1.88072   Max.   :4.2529
(Other)           :374
```

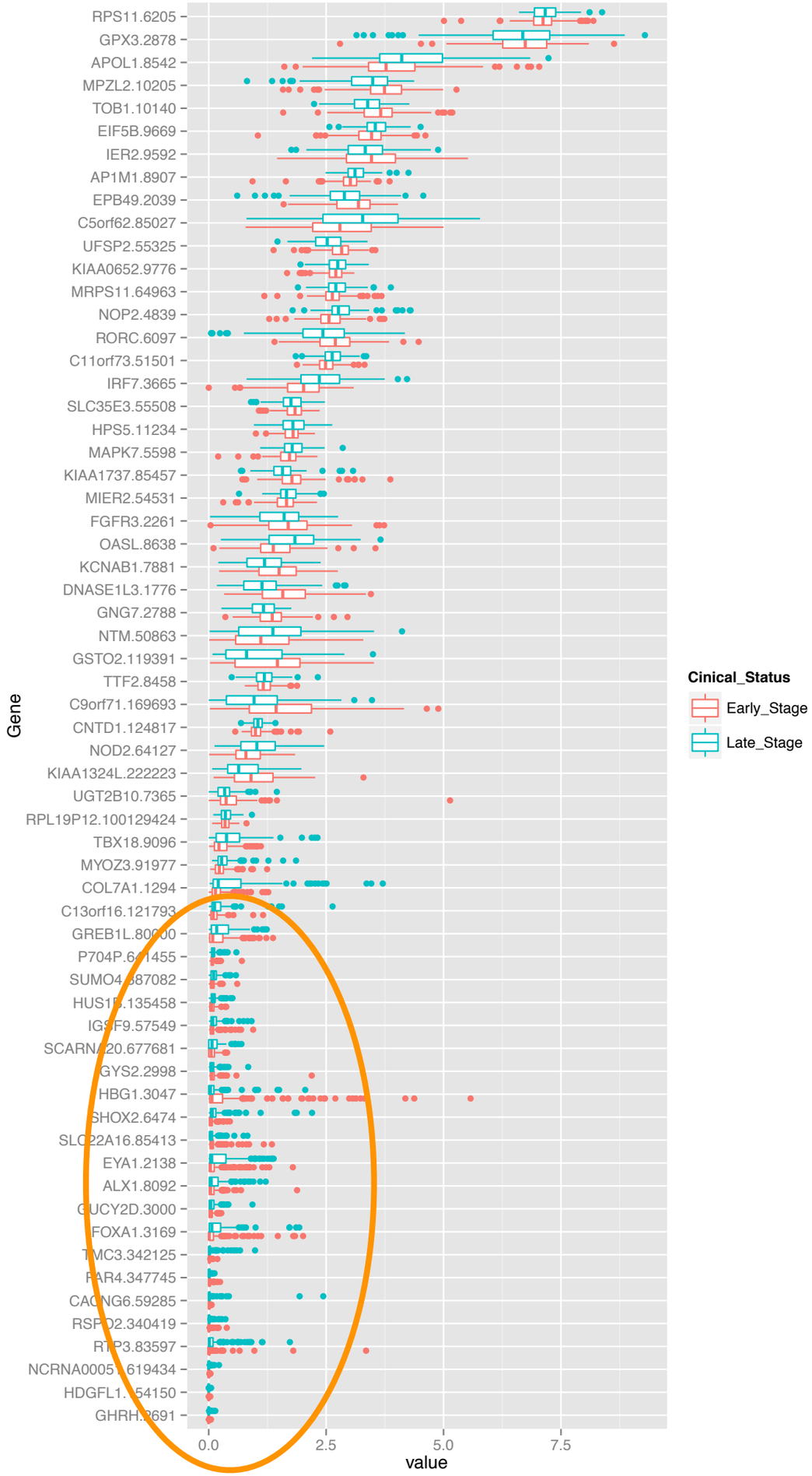
Take a look

- Two groups, 62 variables

- Make side-by-side boxplots of log counts for each gene across all patients, separately for each cancer stage

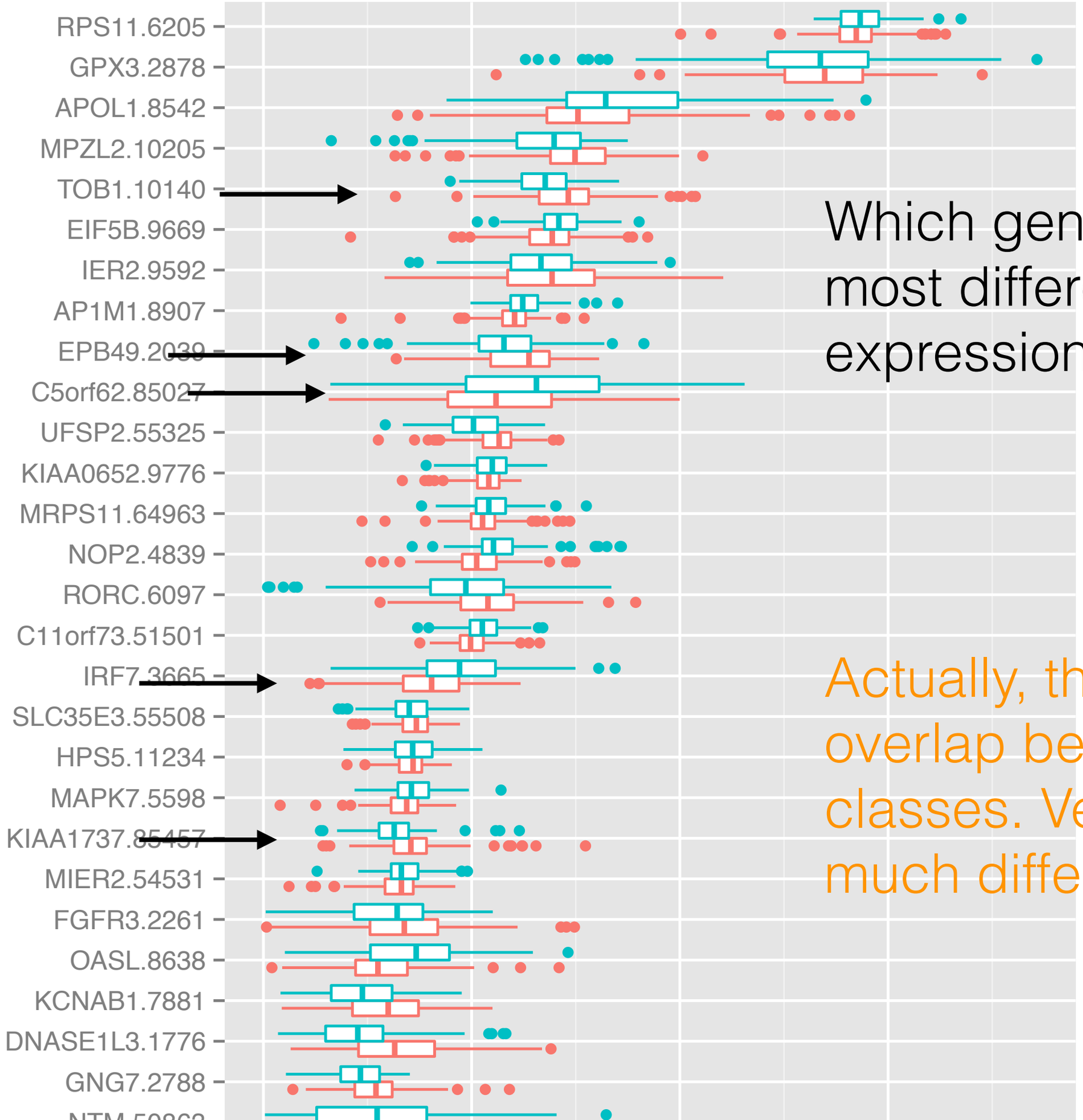
  - Order genes by median

```
> tr.s.g <- gather(tr.s, Gene, value, -Patient_id, -Cinical_Status)
> gm <- summarise(group_by(tr.s.g, Gene), m = median(value))
> tr.s.g$Gene <- factor(tr.s.g$Gene, levels=gm$Gene[order(gm$m)])
> qplot(Gene, value, data=tr.s.g, color=Cinical_Status,
    geom="boxplot") + coord_flip()
```



Mostly interested to see which of the genes have different expression values on early vs late stage

Main thing we notice is that some of the genes, actually a lot have very low values. This should make us suspicious about the filtering used.



Which genes have the most difference in expression values?

Actually, there's a lot of overlap between the classes. Very few with much difference.

Your Turn

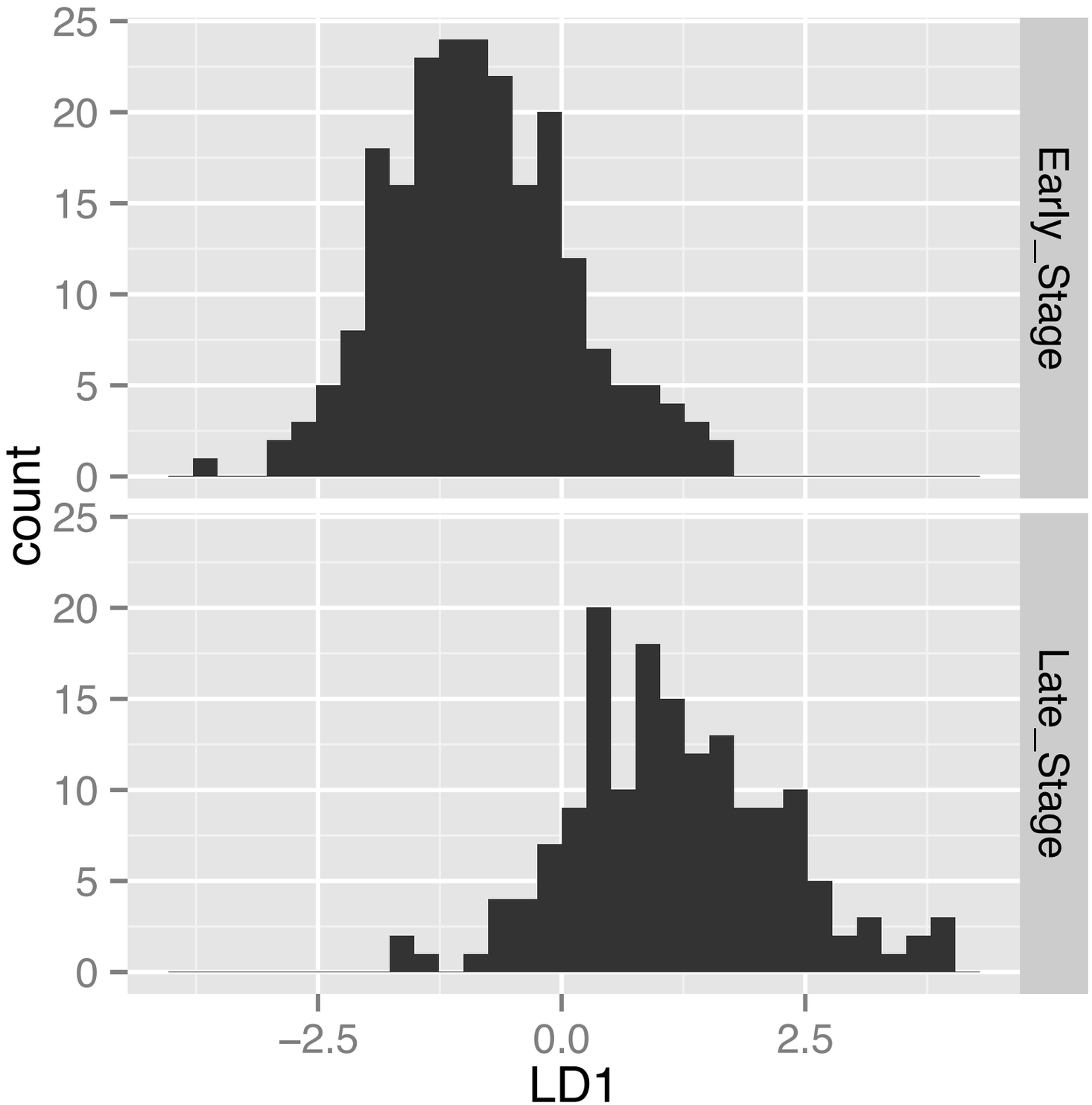
Given that the main purpose is to explore differential expression between the two cancer stages, how might you tweak this plot a little more?

Linear Combinations

- Examine linear combinations of the genes to find a better distinction between the two cancer stages

- Something simple, linear discriminant analysis

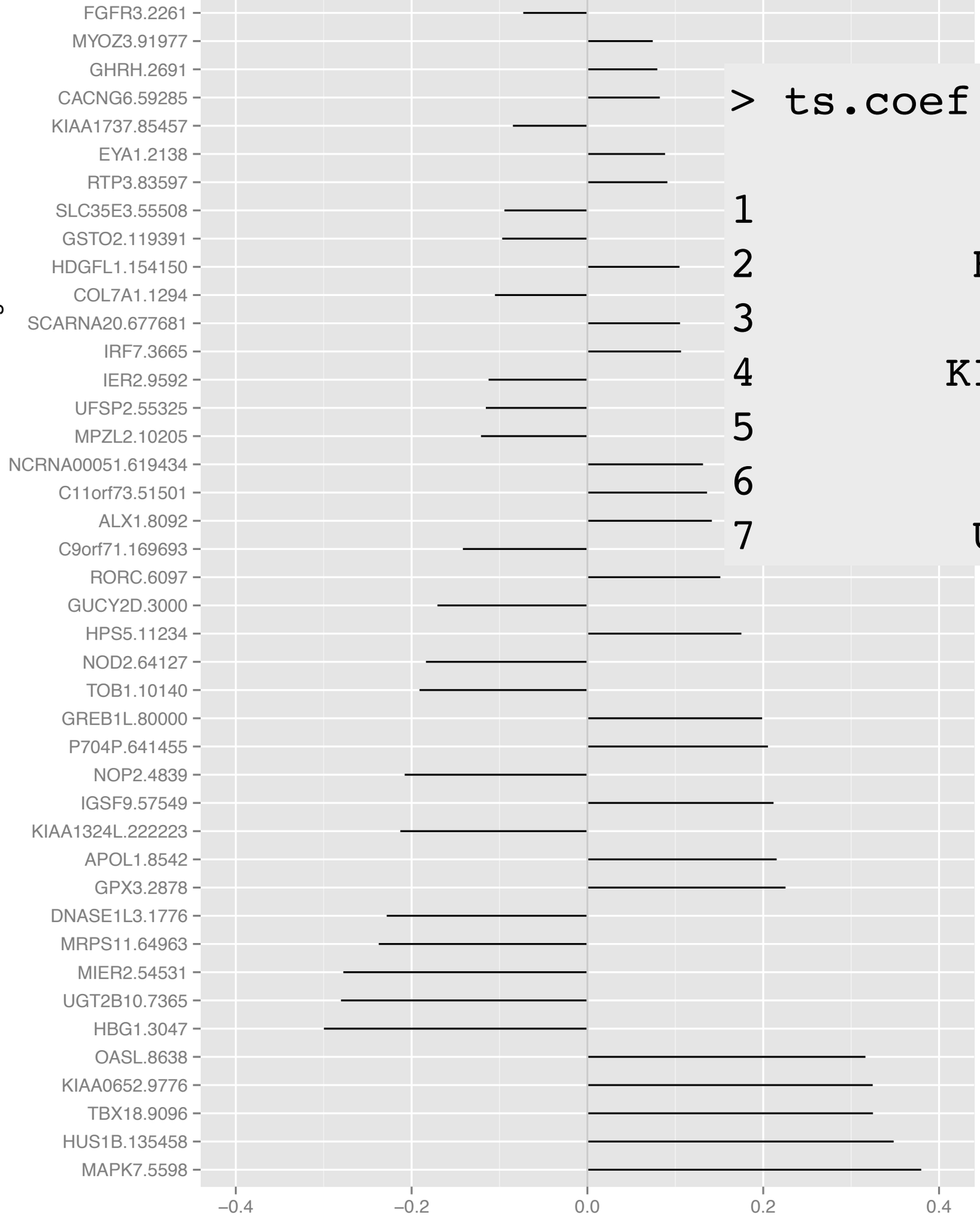
```
> library(MASS)
> # Standardize all the counts
> tr.s.sd <- data.frame(Patient_id=tr.s[,1],
    scale(tr.s[,2:63]), Cinical_Status=tr.s[,64])
> tr.lda <- lda(Cinical_Status~., data=tr.s.sd[,-1])
> tr.p <- predict(tr.lda, tr.s.sd)
> table(tr.s.sd$Cinical_Status, tr.p$class)

              Early_Stage Late_Stage
  Early_Stage         196         24
  Late_Stage           31        129
> tr.df <- data.frame(class=tr.s.sd$Cinical_Status, tr.p$x)
> qplot(LD1, data=tr.df, geom="histogram", facets=class~.)
```



Now, it looks like there is more of a difference

Which genes matter?

```
> #Plot the coefficients, which genes contribute the most?
> ts.coef <- data.frame(g=rownames(tr.lda$scaling),
    tr.lda$scaling)
> ts.coef$ymin <- ifelse(ts.coef$LD1<0, ts.coef$LD1, 0)
> ts.coef$ymax <- ifelse(ts.coef$LD1>0, ts.coef$LD1, 0)
> ts.coef <- arrange(ts.coef, desc(abs(ts.coef$LD1)))
> ts.coef$g <- factor(ts.coef$g, levels=ts.coef$g)
> ggplot(ts.coef) + geom_linerange(aes(x=g,
      ymin=ymin, ymax=ymax)) +
    geom_hline(yintercept=0, color="grey80") +
      ylim(-0.4, 0.4) +
    coord_flip()
```

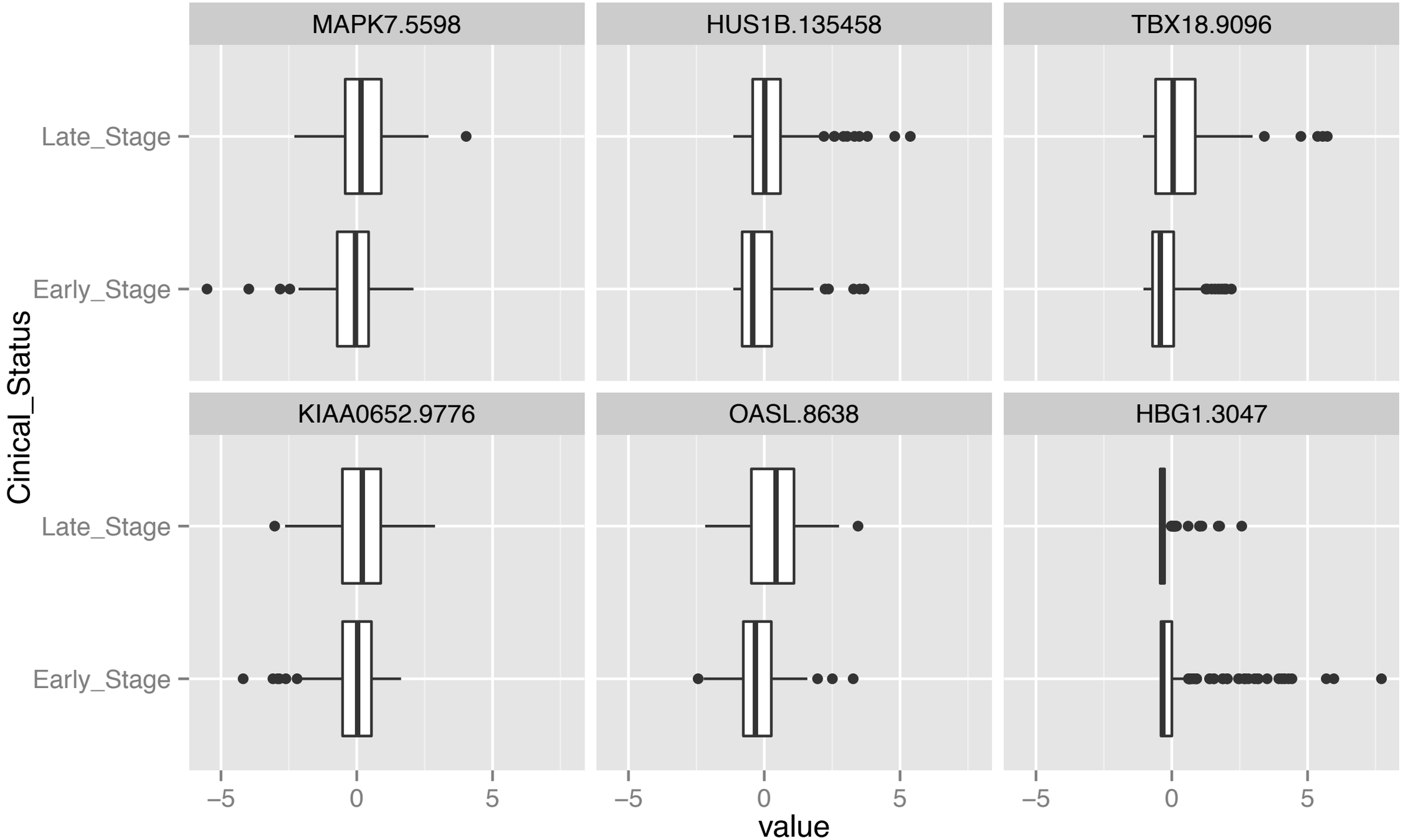



```
> ts.coef[,1:2]
                          g       LD1
1          MAPK7.5598  0.37967
2        HUS1B.135458  0.34835
3          TBX18.9096  0.32485
4       KIAA0652.9776  0.32453
5           OASL.8638  0.31627
6           HBG1.3047 -0.29974
7        UGT2B10.7365 -0.28030
```

Slow decline in magnitude

Most important genes

```
> # Plot the most important genes
> tr.s.sd.g <- gather(tr.s.sd, Gene, value, -Patient_id,
    -Cinical_Status)
> g1 <- filter(tr.s.sd.g, Gene %in% ts.coef[1:6,1])
> g1$Gene <- factor(g1$Gene, levels=ts.coef[1:6,1])
> qplot(Cinical_Status, value, data=g1, geom="boxplot") +
  facet_wrap(~Gene)
```

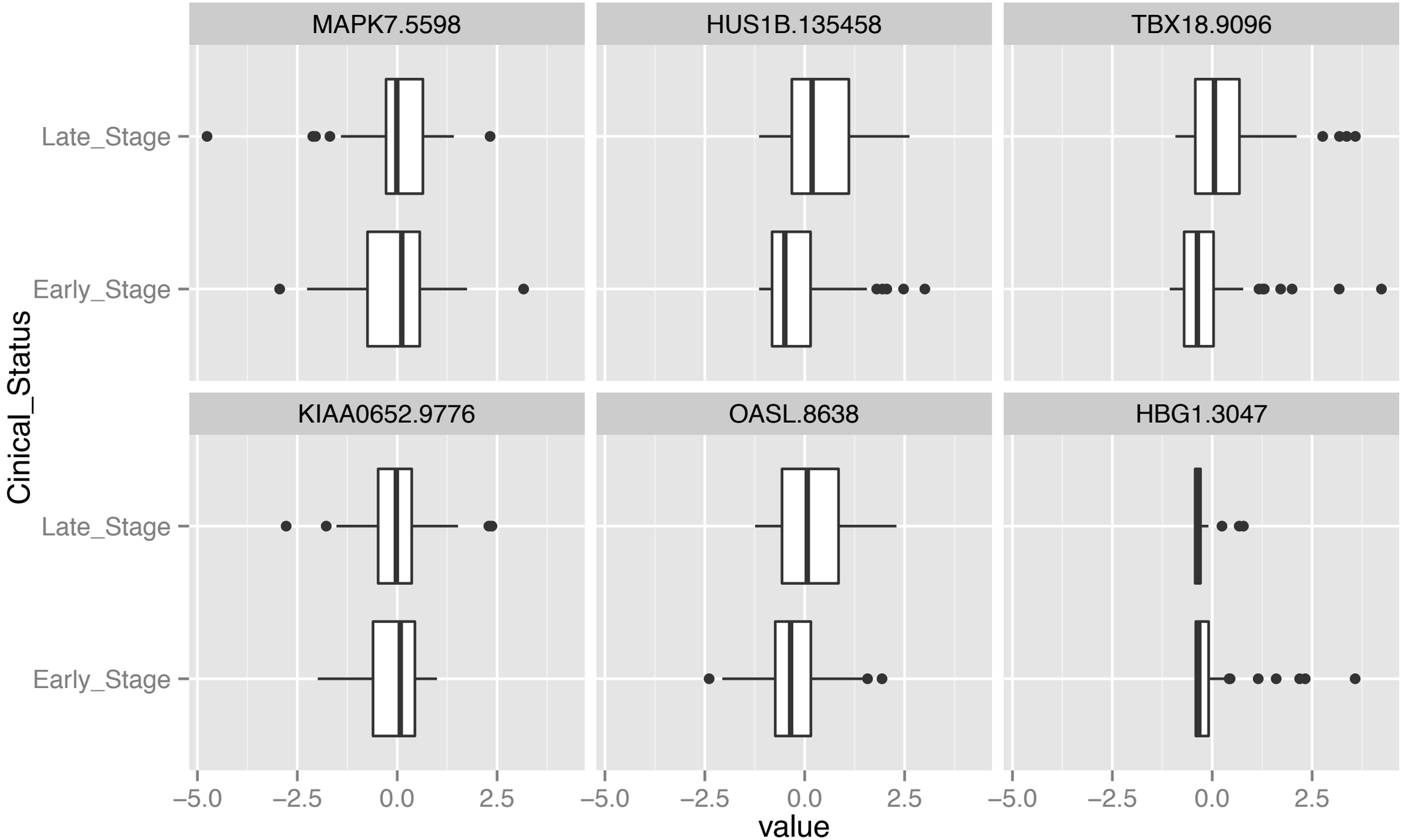

MAPK7.5598
HUS1B.135458
TBX18.9096
KIAA0652.9776
OASL.8638
HBG1.3047
Late_Stage
Early_Stage
Cinical_Status
−5
0
5
value

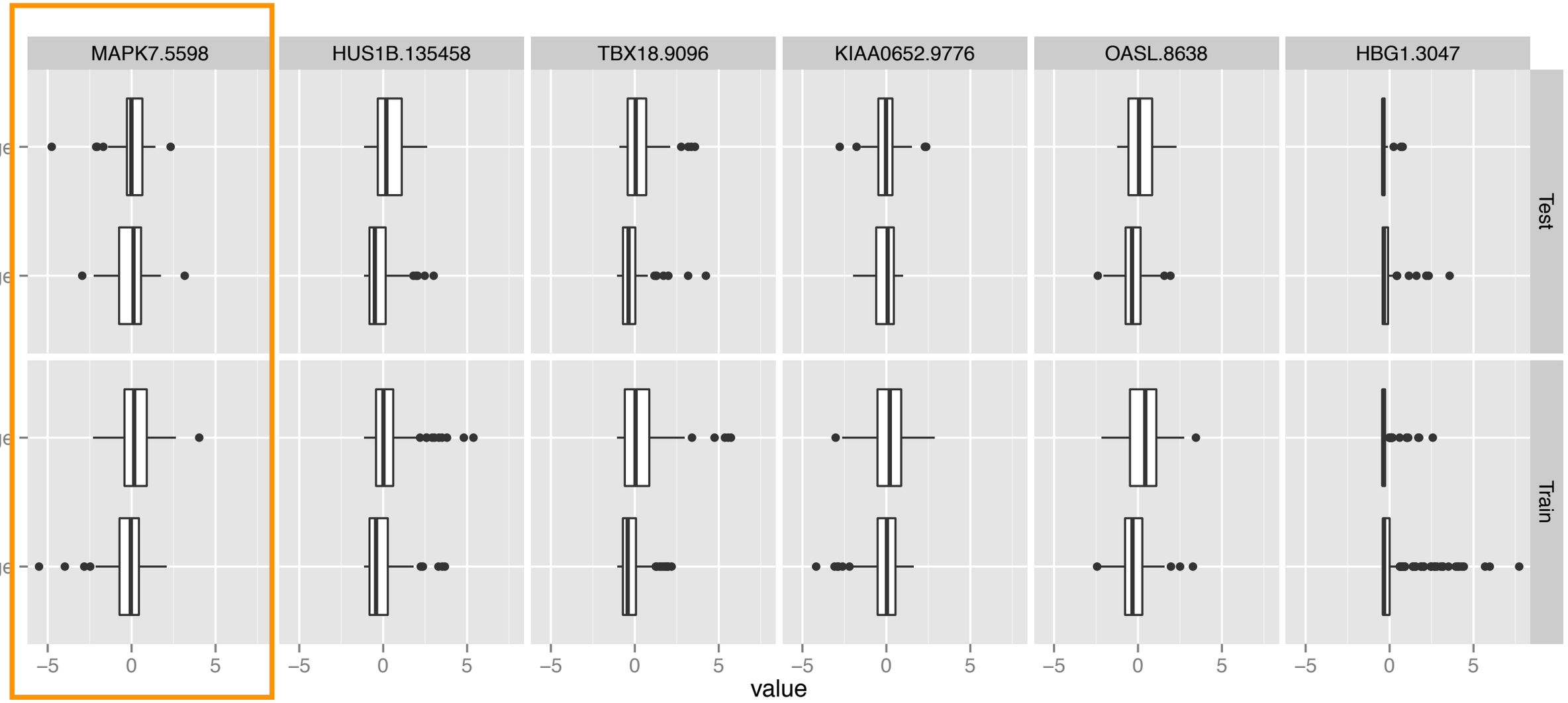
Check the test data

- Training error should be small - it is 14%
- It is important to see how the model works for the data that was not used to build the model, the test set.
- Compare the plots of important variables
- Predict the test set, examine the error

Plot the test data

```
> tr.s.sd.g <- gather(tr.s.sd[,2:64], key="Cinical_Status")
> colnames(tr.s.sd.g)[2] <- "Gene"
> g2 <- filter(ts.s.sd.g, Gene %in% ts.coef[1:6,1])
> g2$Gene <- factor(g2$Gene, levels=ts.coef[1:6,1])
> qplot(Cinical_Status, value, data=g2, geom="boxplot") +
  facet_wrap(~Gene) + coord_flip()
```


MAPK7.5598
HUS1B.135458
TBX18.9096
KIAA0652.9776
OASL.8638
HBG1.3047
Late_Stage
Early_Stage
Cinical_Status
−5.0
−2.5
0.0
2.5
value



Distributions should be the same

```
> g <- rbind(g1, g2)
> g$set <- c(rep("Train", 2280), rep("Test", 570))
> qplot(Cinical_Status, value, data=g, geom="boxplot") +
    facet_grid(set~Gene) + coord_flip()
```

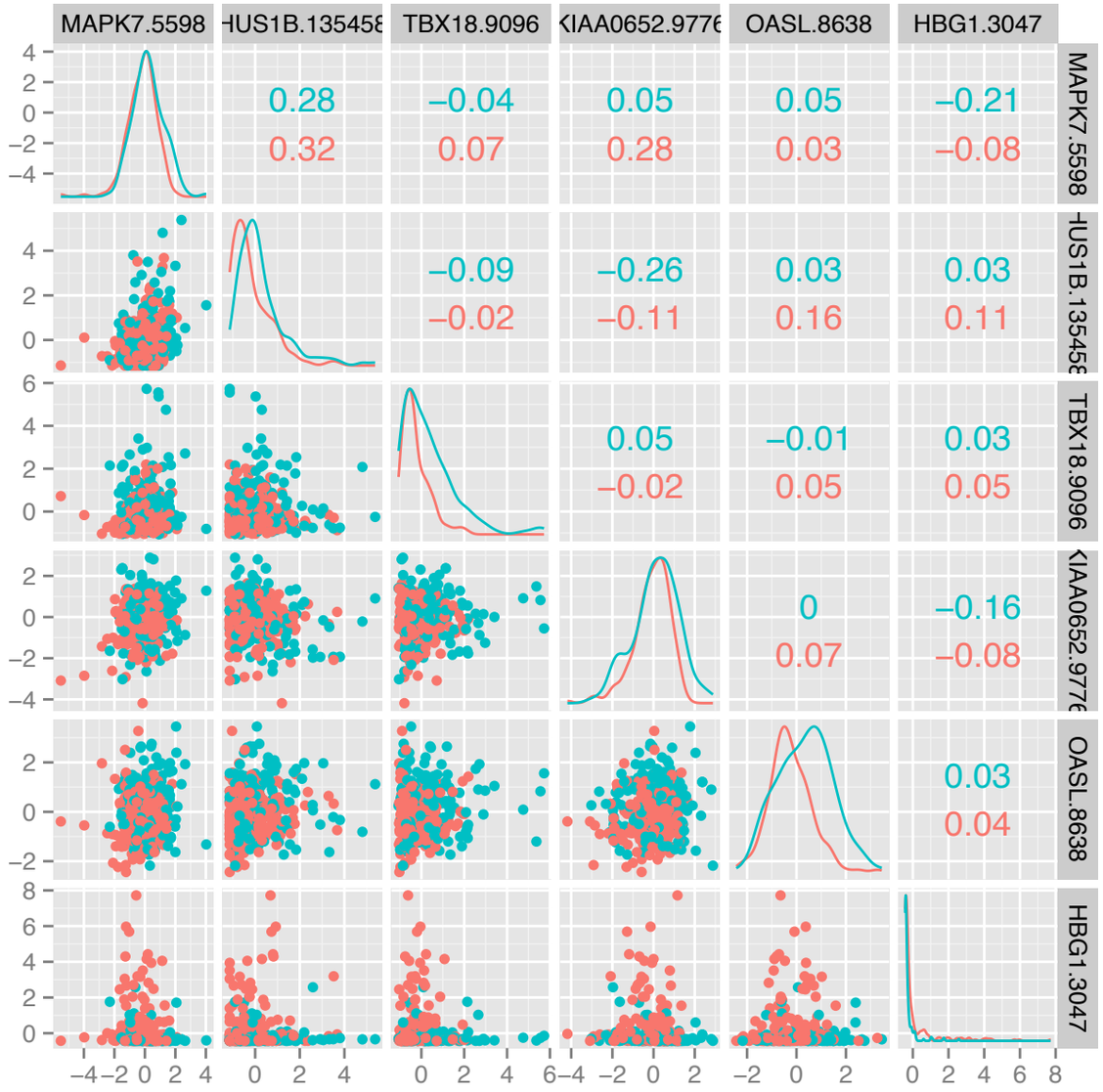
Pairwise view

- Make a scatterplot matrix of the 6 most important variables

- Separately for training and test

```
# Scatterplot matrix
> g1.s <- spread(g1, Gene, value)
> ggscatmat(g1.s, columns=3:8, color=“Cinical_Status")
> g2.s <- spread(g2, Gene, value)
> ggscatmat(g2.s, columns=3:8, color="Cinical_Status")
```

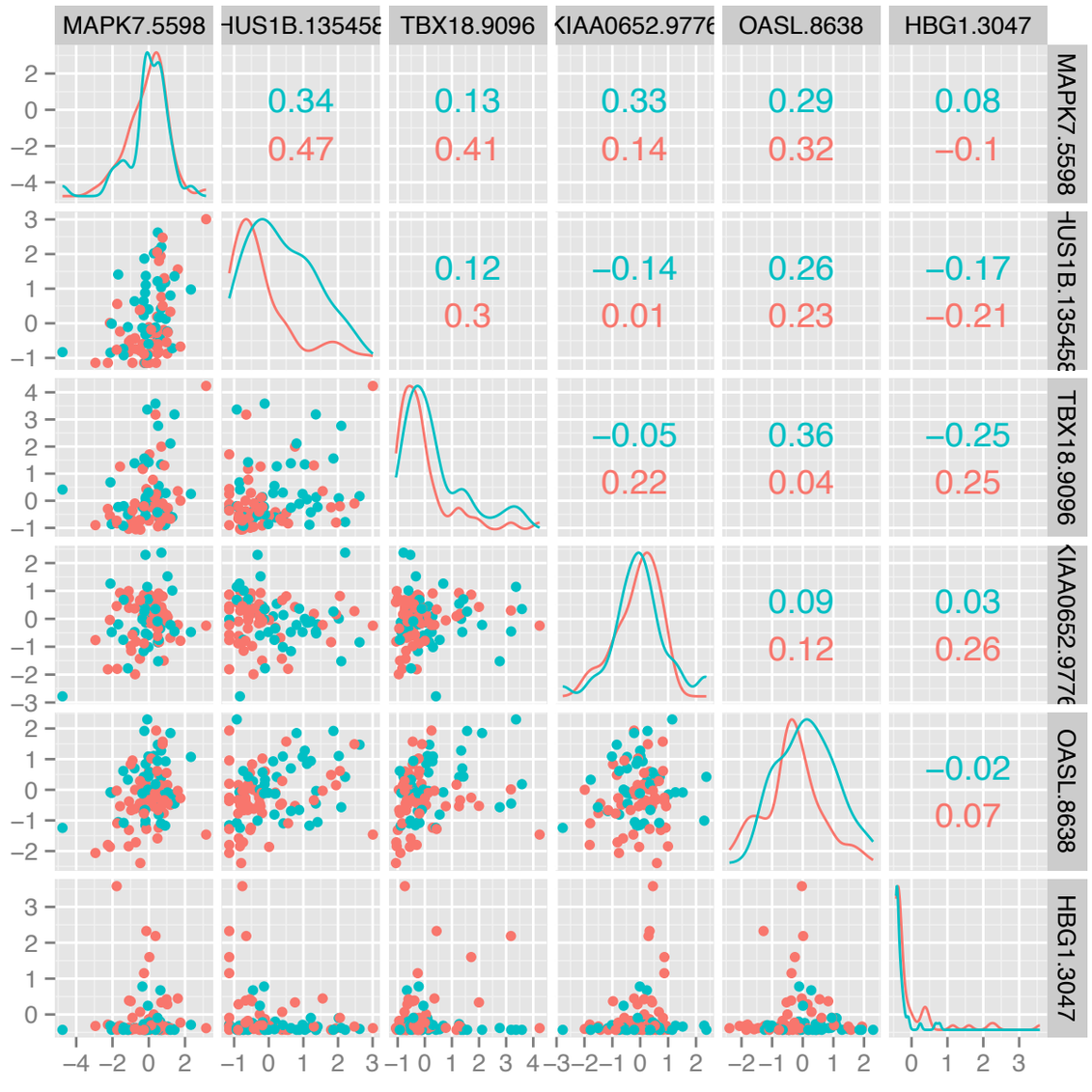
Training



Cinical_Status

- Early_Stage
- Late_Stage

Test



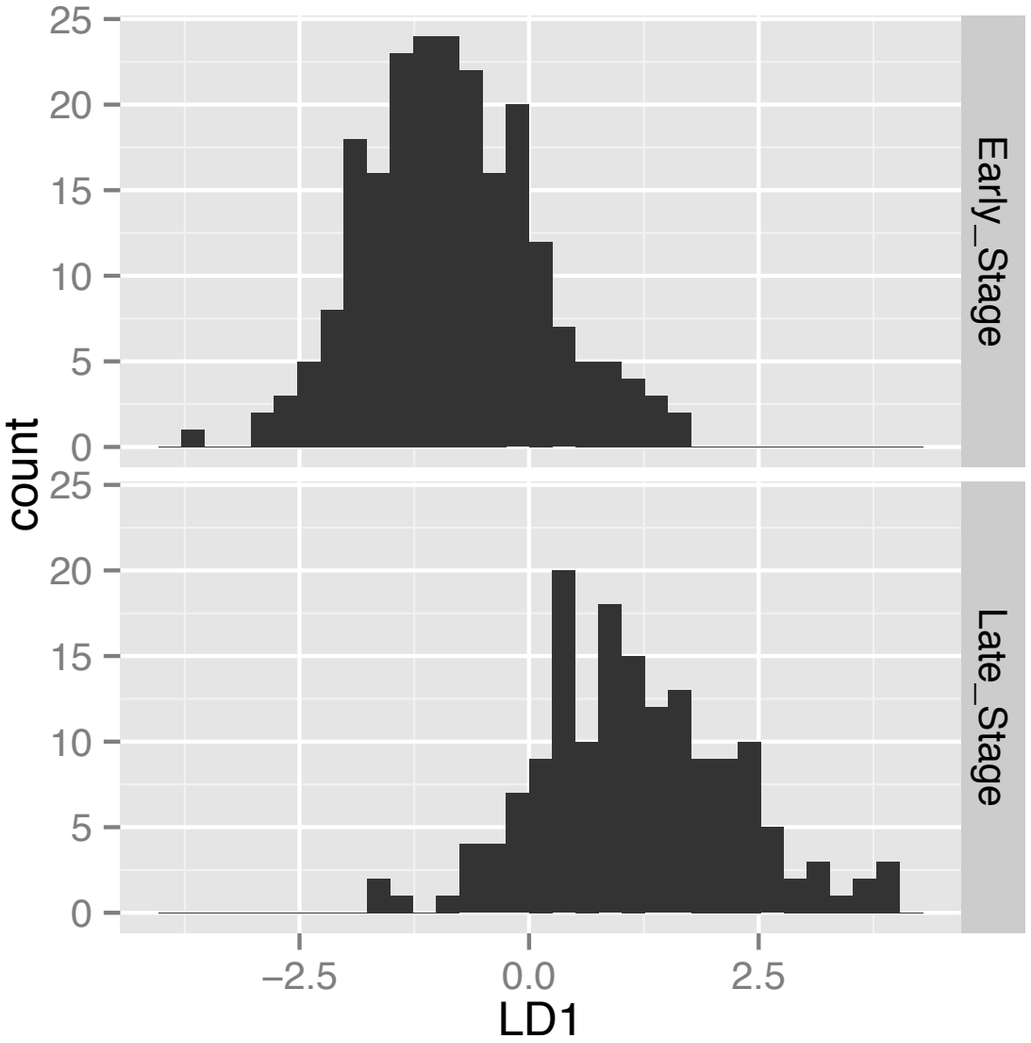
- Do they look the same?
- Are the two stages as differentiated in the test as training?

Predict the test

- Using the model built on the training data, predict the test set
- Errors should be similar, maybe a little worse for the test data
- Projected data should look similar to the training data

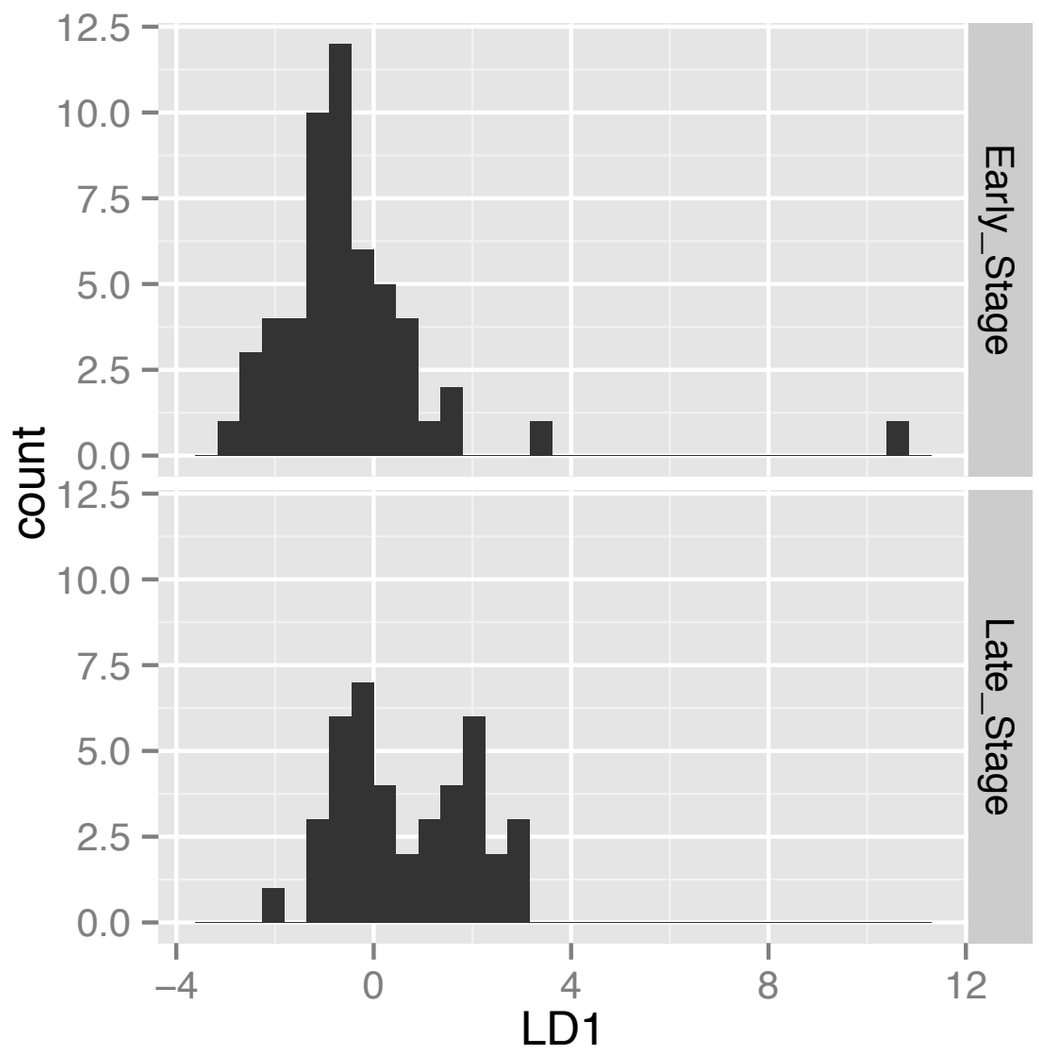
```
> ts.p <- predict(tr.lda, ts.s.sd)
> table(ts.s.sd$Cinical_Status, ts.p$class)

              Early_Stage Late_Stage
  Early_Stage          45          9
  Late_Stage           20         21
> ts.df <- data.frame(class=ts.p$class, ts.p$x)
> qplot(LD1, data=ts.df, geom="histogram", facets=class~.)
```

Training



Error 14%

Test



Error 31%

Two clues to problems. What are they?

Your Turn

- Read in the challenge-training data.
- This is similar data to the renal cancer data set, with essentially the same variables and classes. The top 60 genes from 20000 have been filtered for you, and the counts have been log-transformed, and standardized.
- Make a side-by-side boxplot of the two cancer stages, by gene.
- Fit the LDA model, predict the class labels and the best projection. Plot the projection as a histogram facetted by class.
- Plot the coefficients.
- Now read the test data, predict it, calculate the error, and make a histogram of the projected values facetted by class.
- What do you learn?

Better classifiers

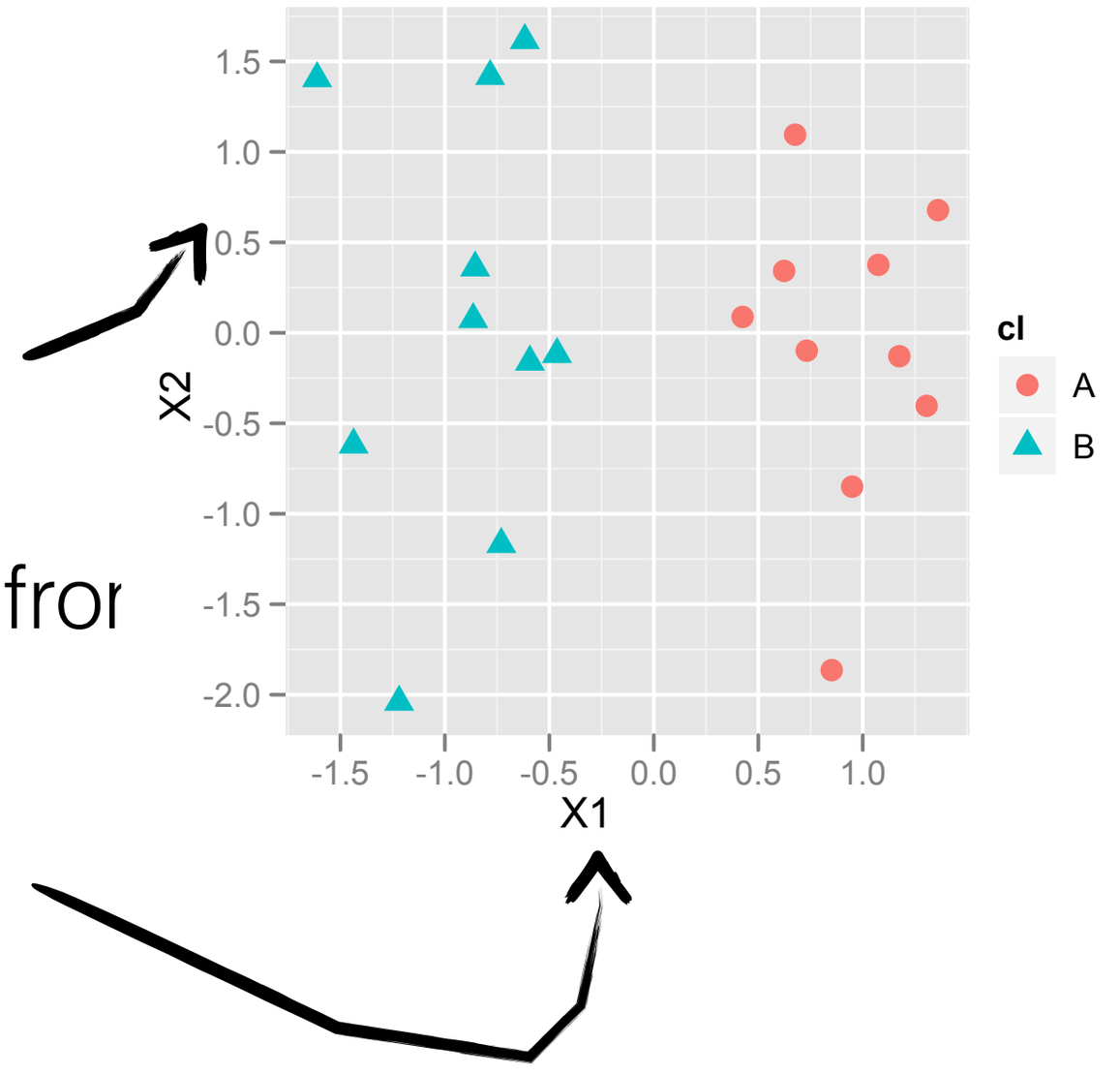
- 20,534 genes and 475 patients is an example of a high-dimension, low sample size data set

- It is possible to have differences which look like differences but are simply that there is not enough samples.

- What is the problem?

Simulated data

- 99 variables simulated from N(0,1)
- One variable simulated half fror N(0,1), and half from N(5,1), standardized
- 20 observations only



Effect of LDA

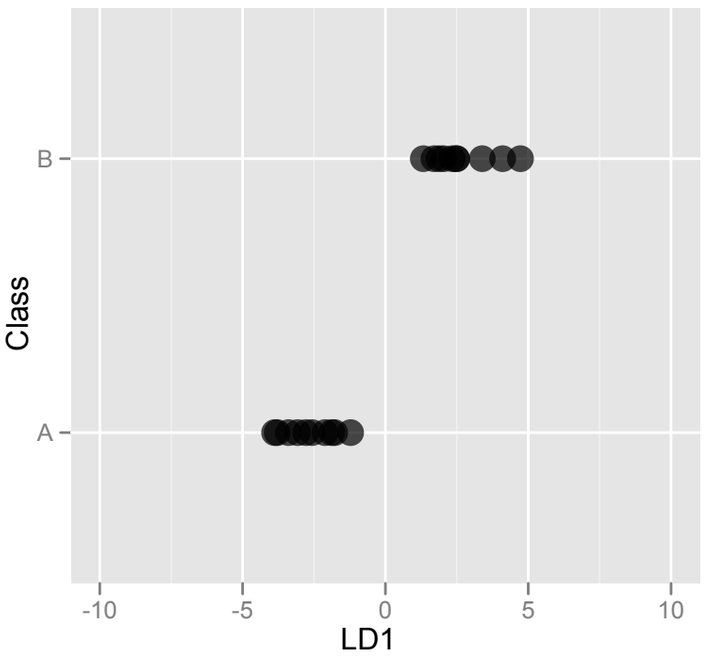
- Fit the LDA with p=2:15
- Examine the
  1. coefficients of the discriminant space
  2. best separation
  3. error rate

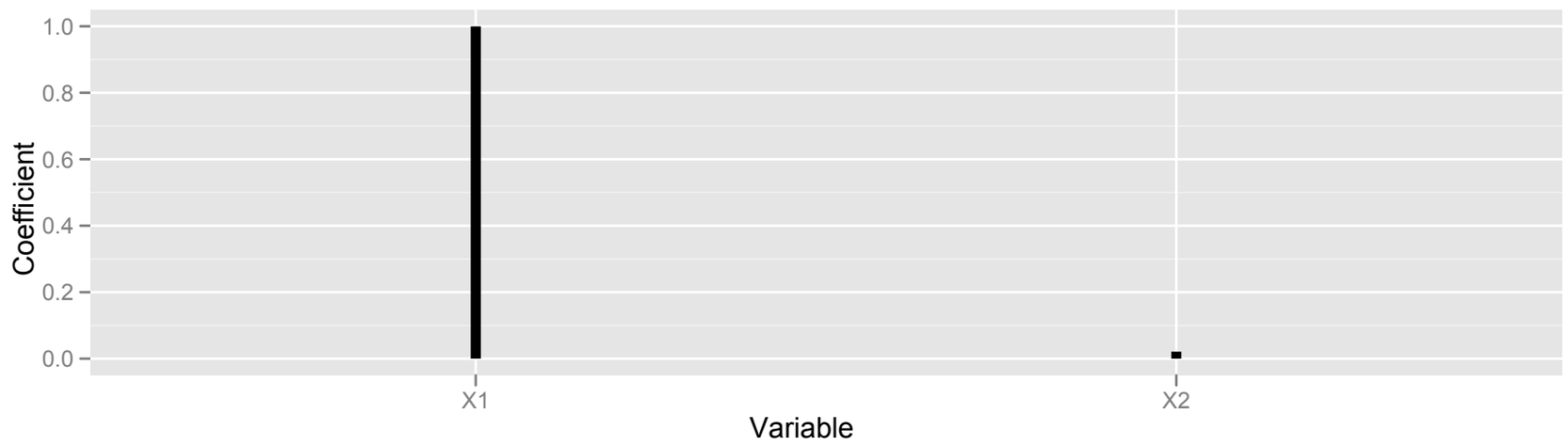
p=2

Good separation

error

	A	B
A	10	0
B	0	10



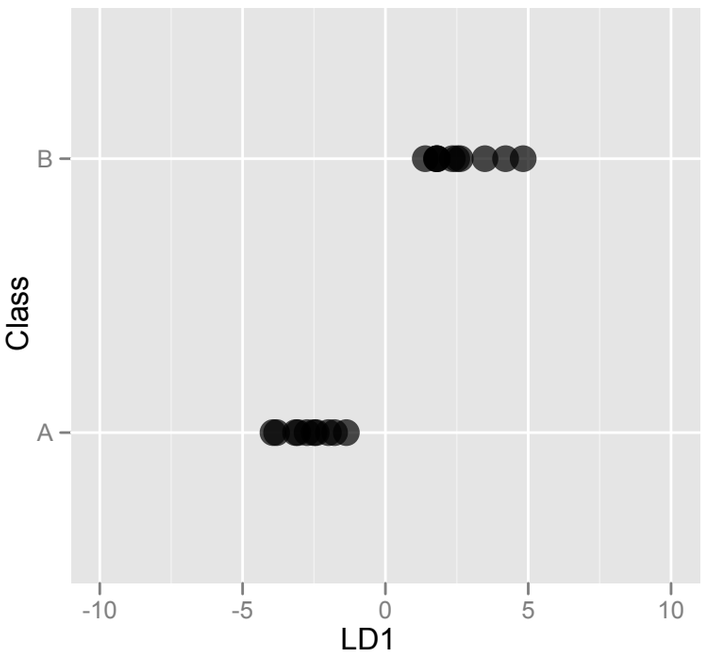


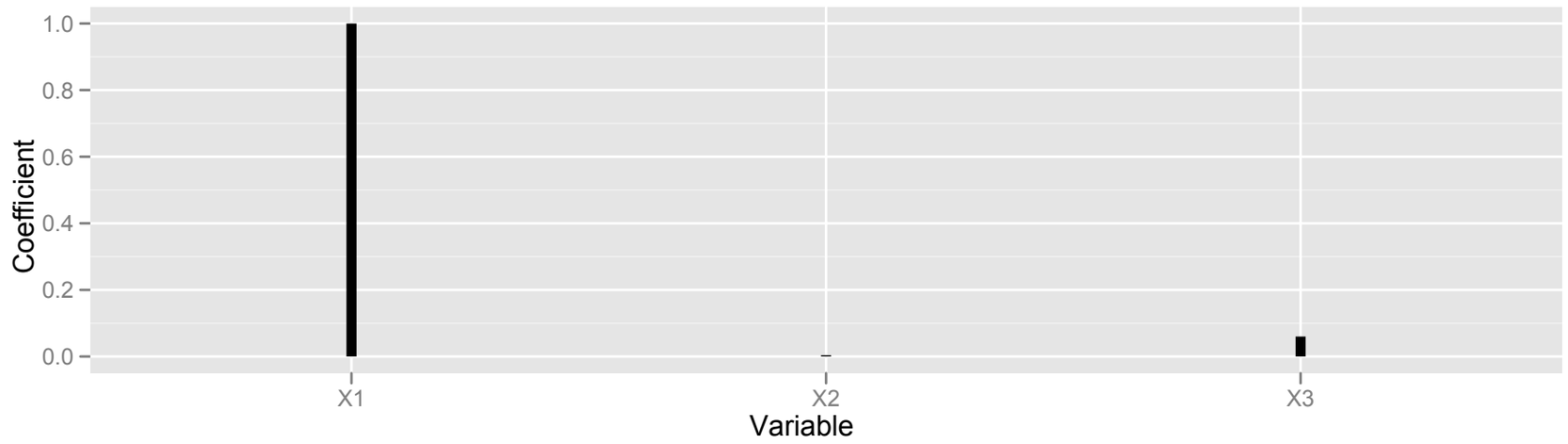
p=3

Good separation

error

	A	B
A	10	0
B	0	10



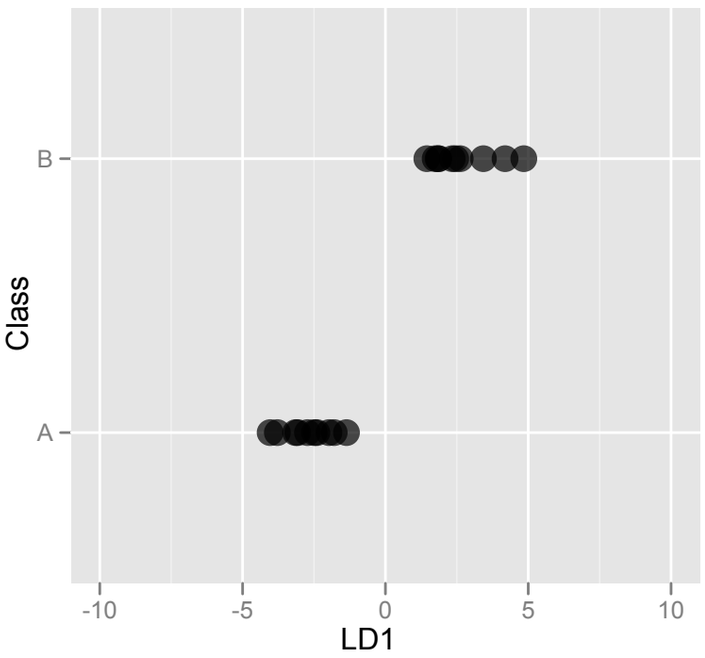


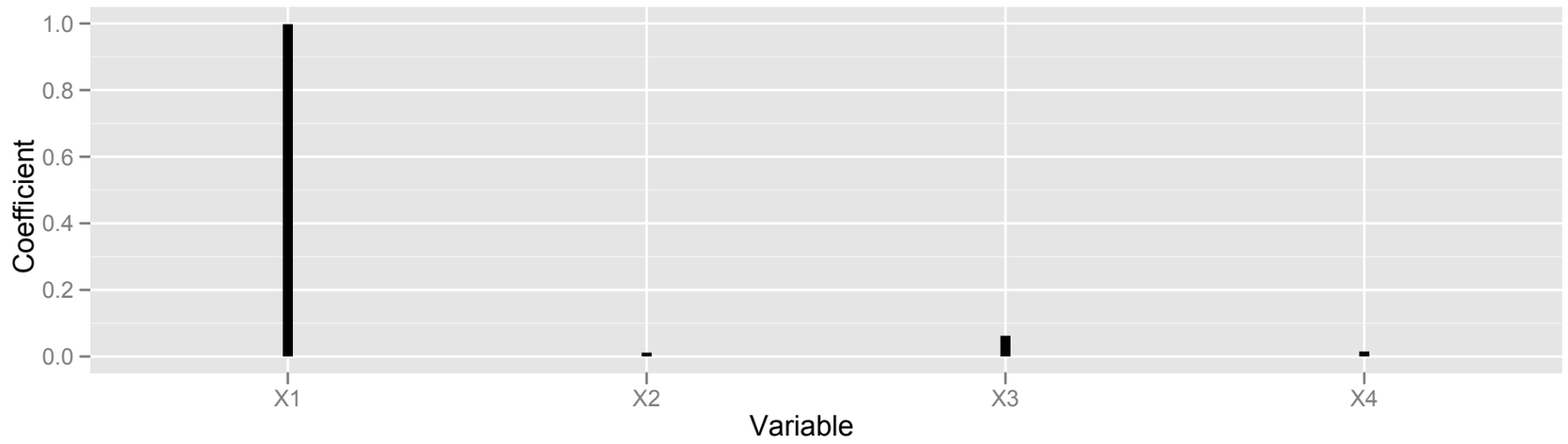
p=4

Good separation

error

	A	B
A	10	0
B	0	10



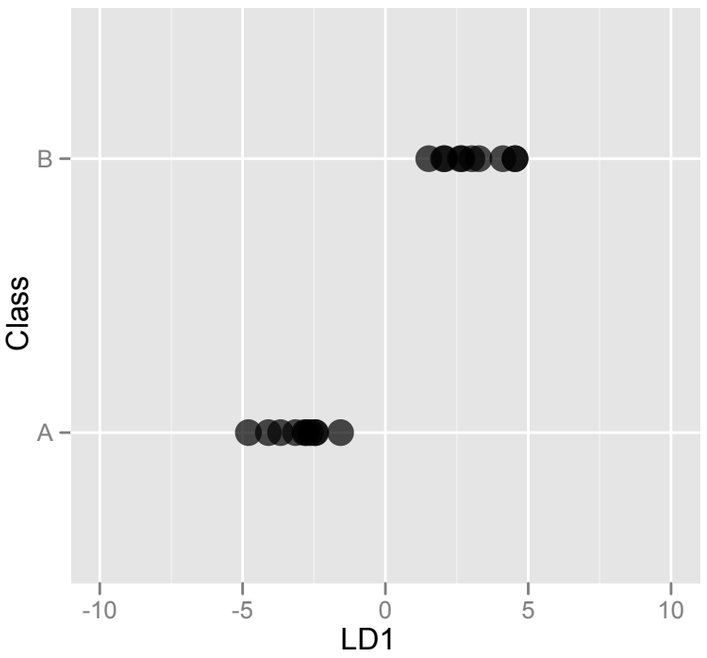


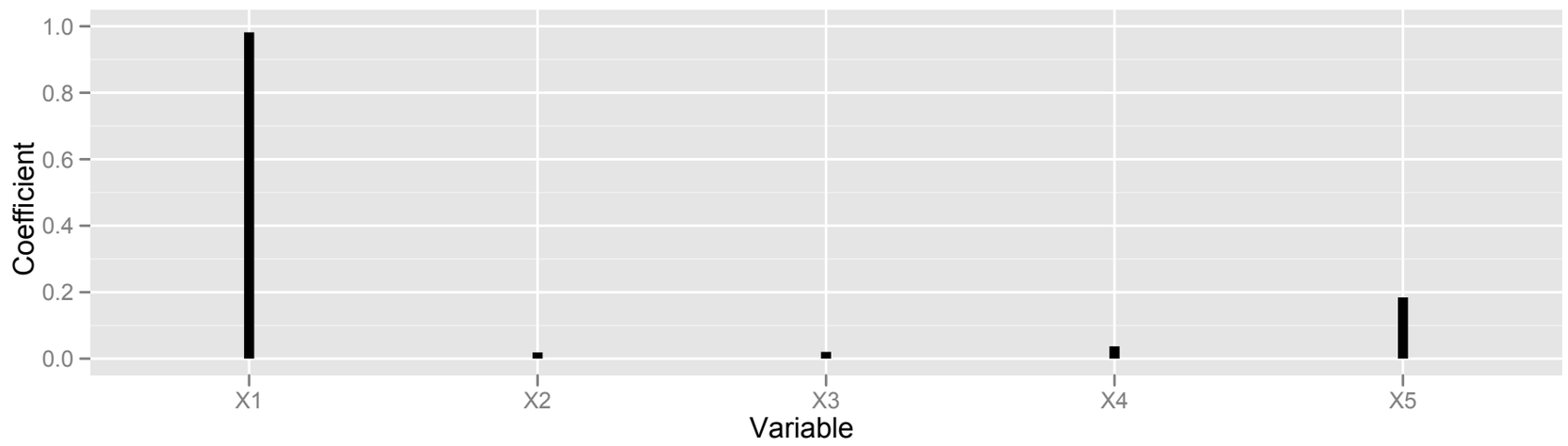
p=5

Good separation

error

	A	B
A	10	0
B	0	10



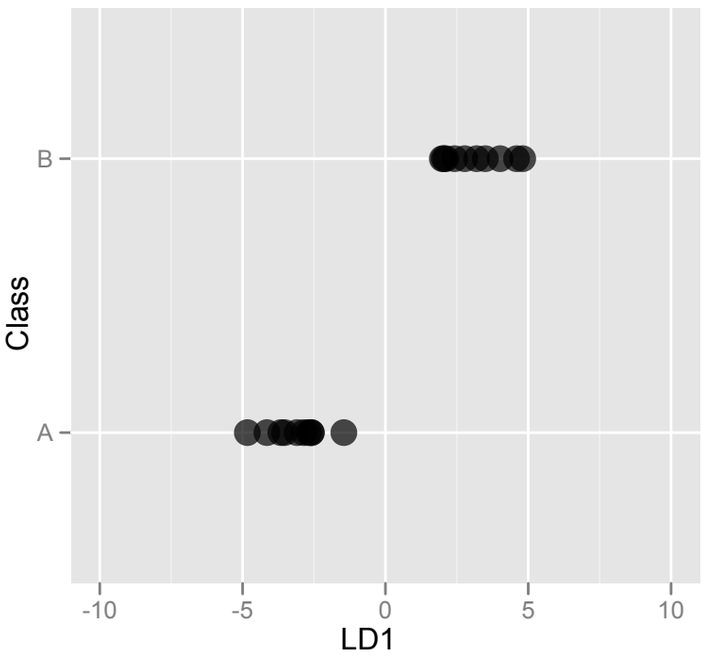


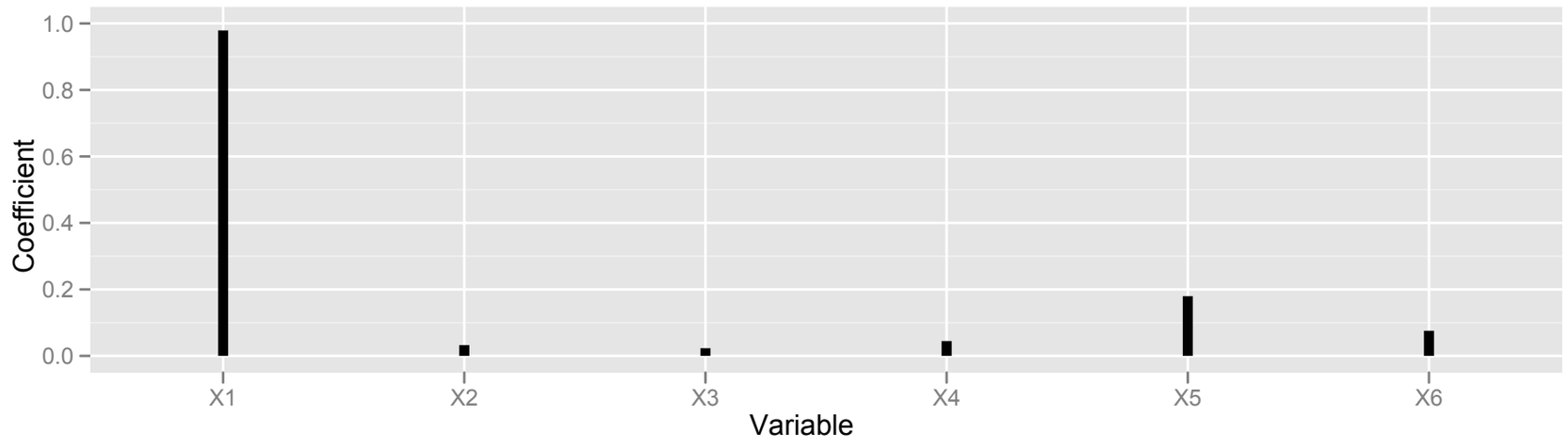
p=6

Good separation

error

	A	B
A	10	0
B	0	10



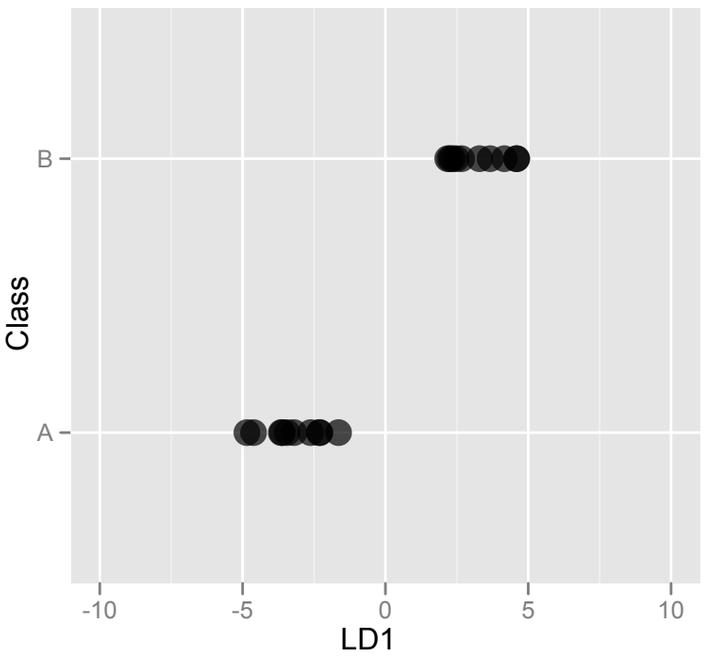


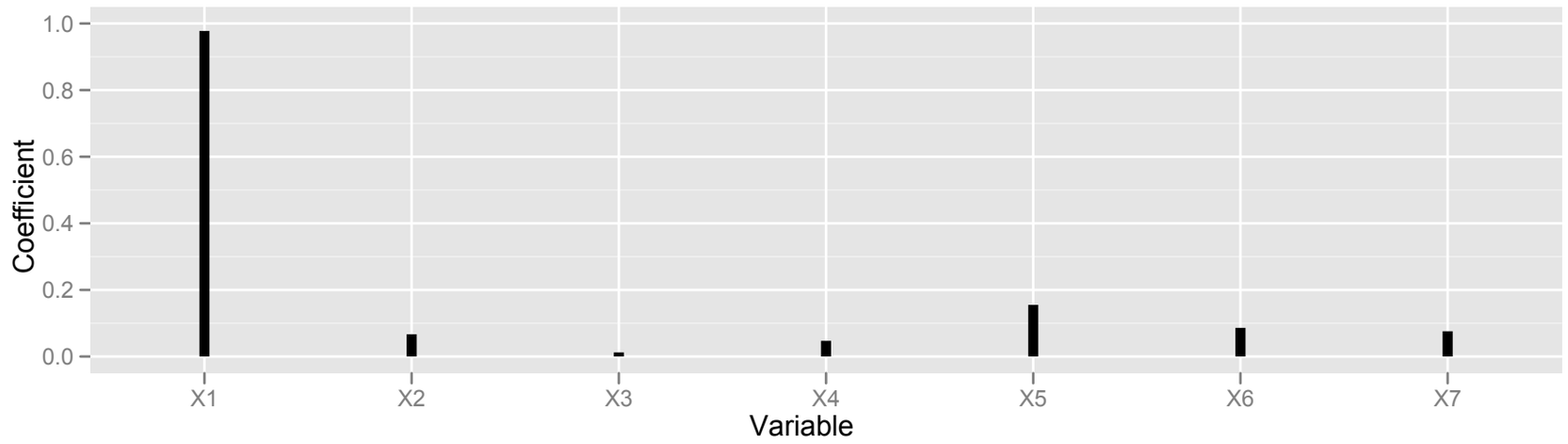
p=7

Good separation

error

	A	B
A	10	0
B	0	10



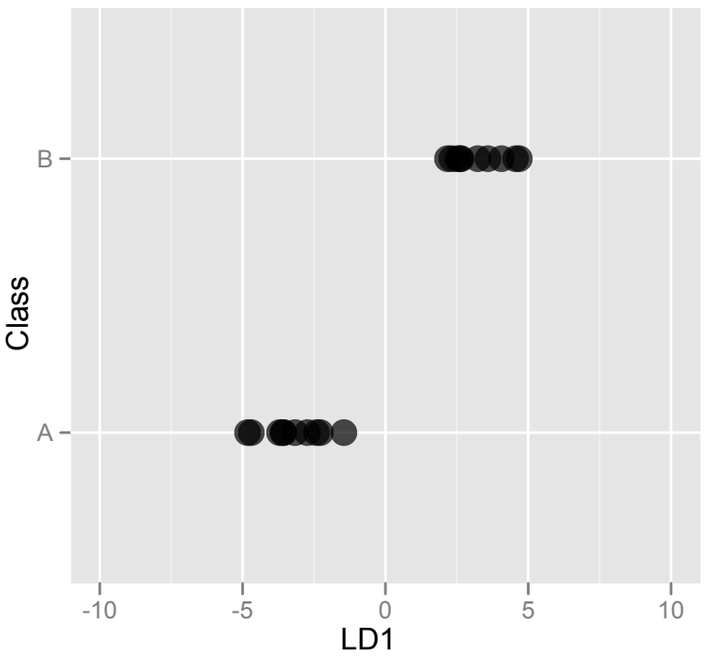


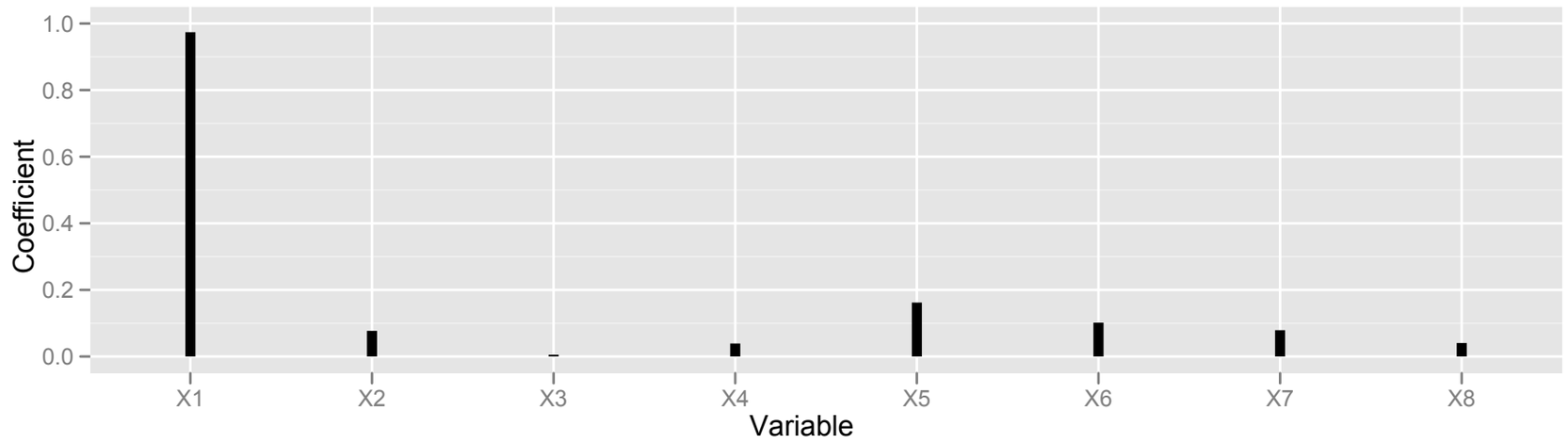
p=8

Good separation

error

	A	B
A	10	0
B	0	10



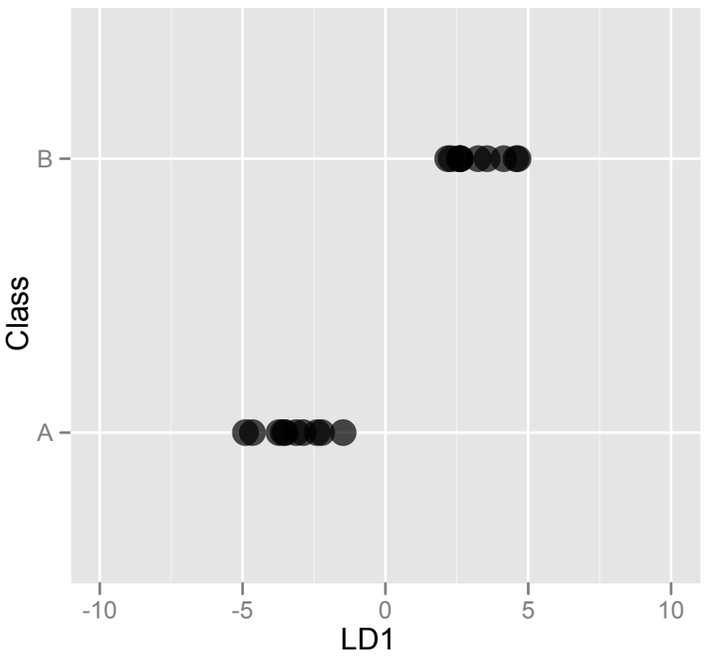


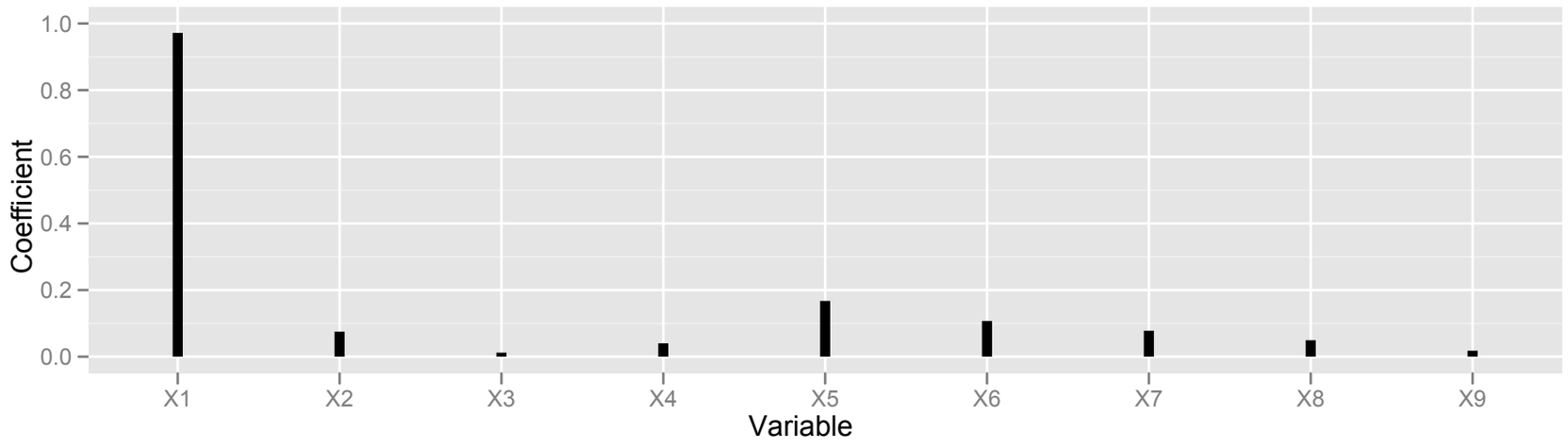
p=9

Good separation

error

	A	B
A	10	0
B	0	10



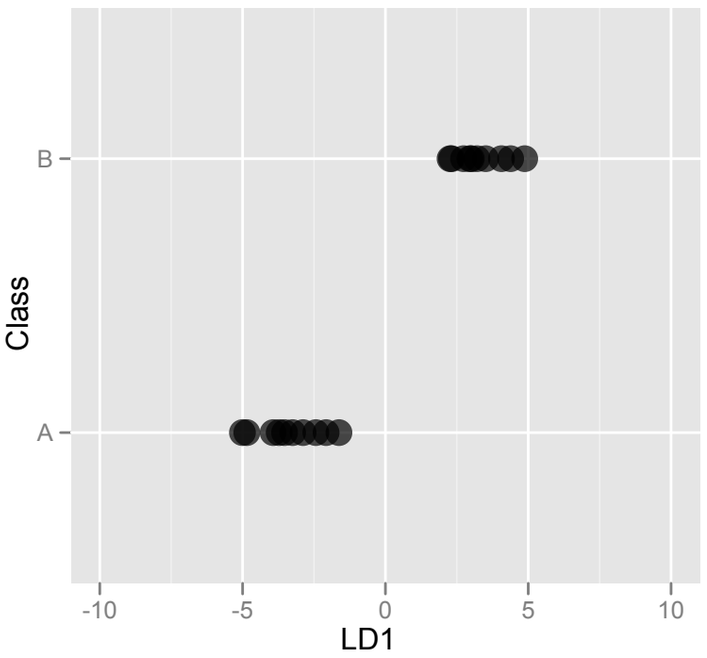


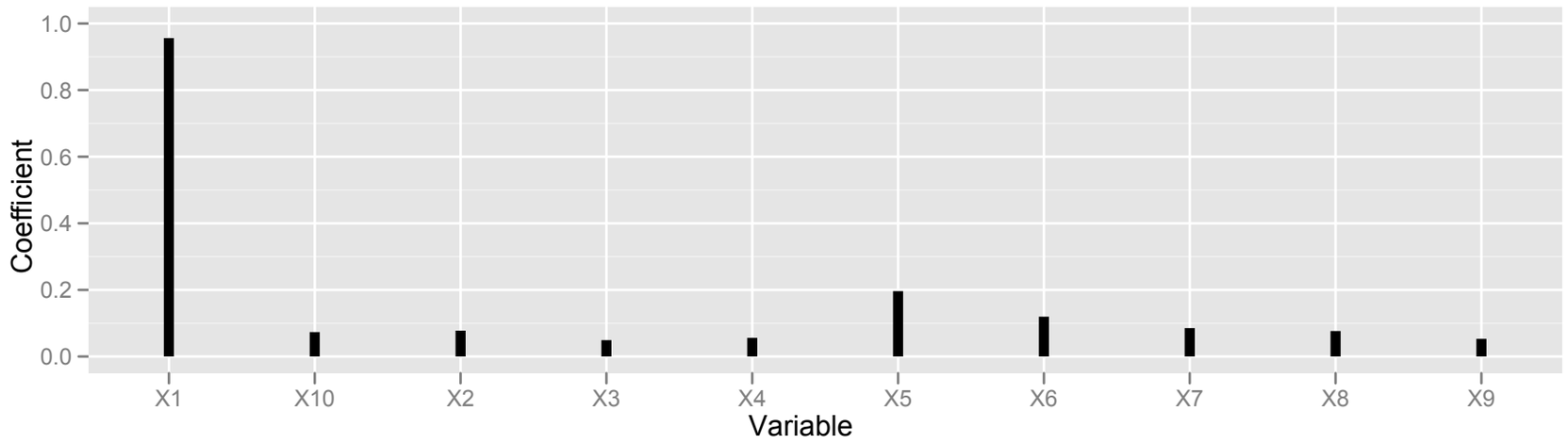
p=10

Good separation

error

	A	B
A	10	0
B	0	10



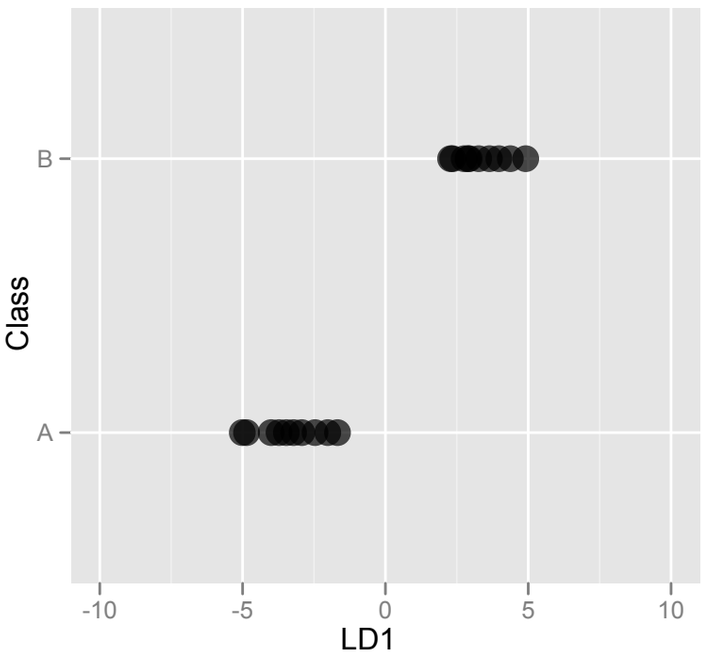


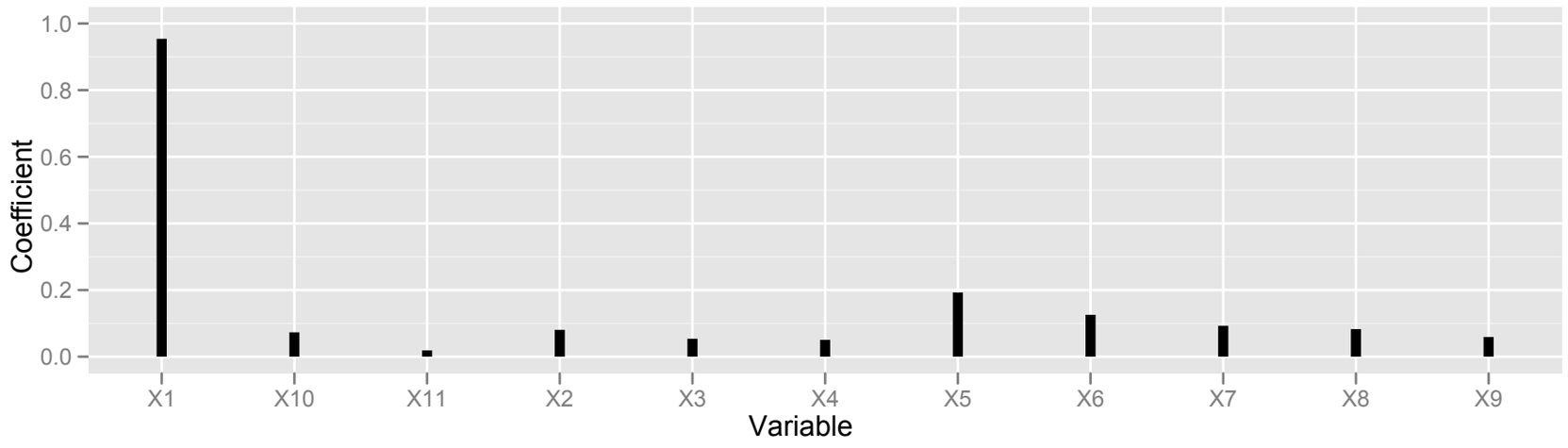
p=11

Good separation

error

	A	B
A	10	0
B	0	10



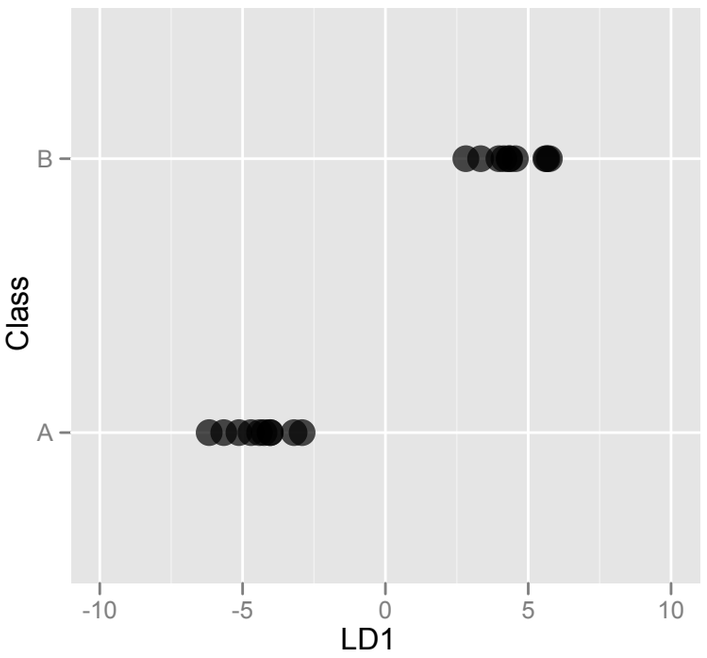


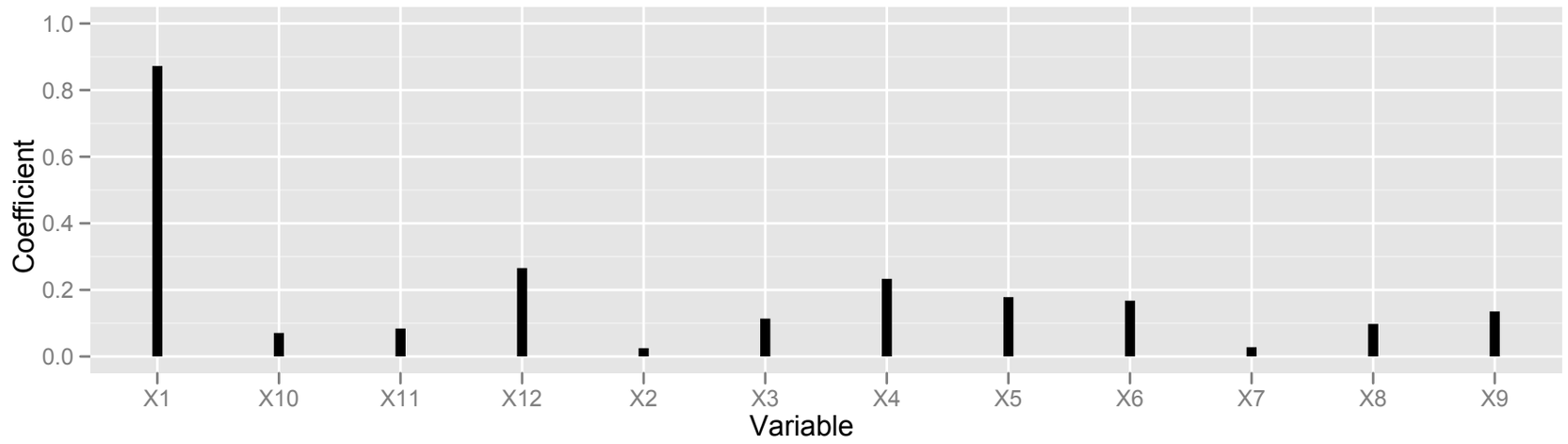
p=12

Good separation

error

	A	B
A	10	0
B	0	10



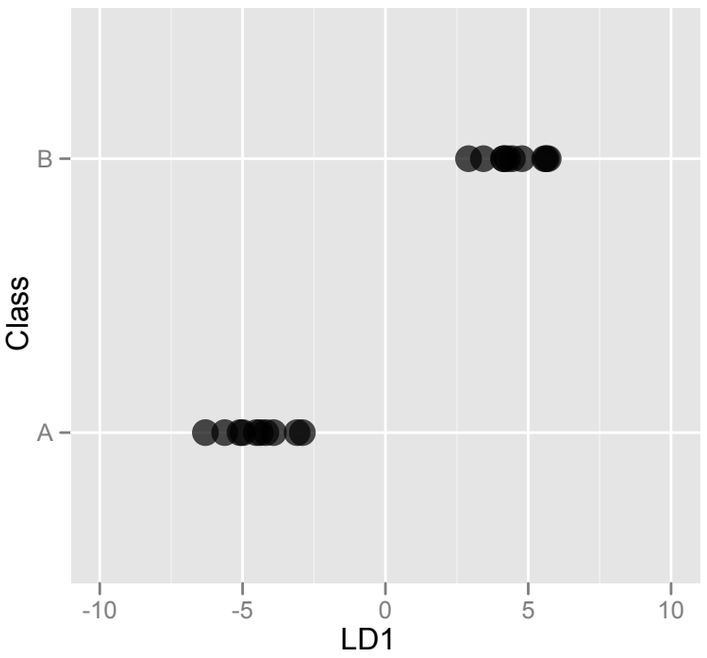


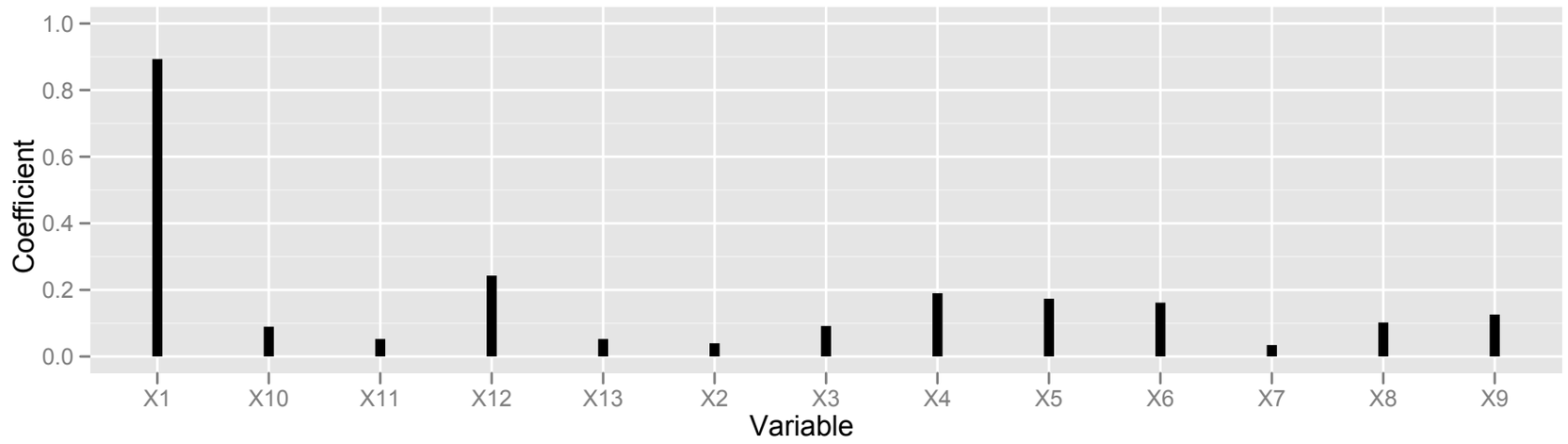
p=13

Good separation

error

	A	B
A	10	0
B	0	10



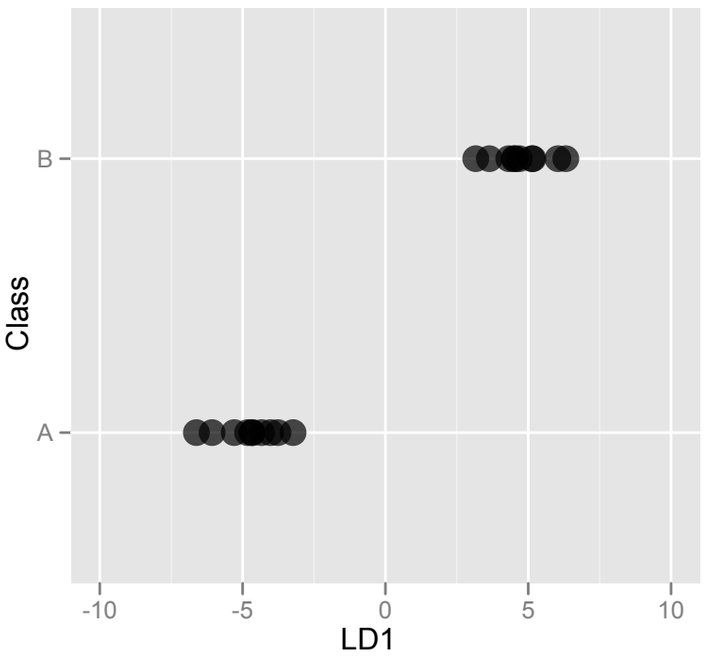


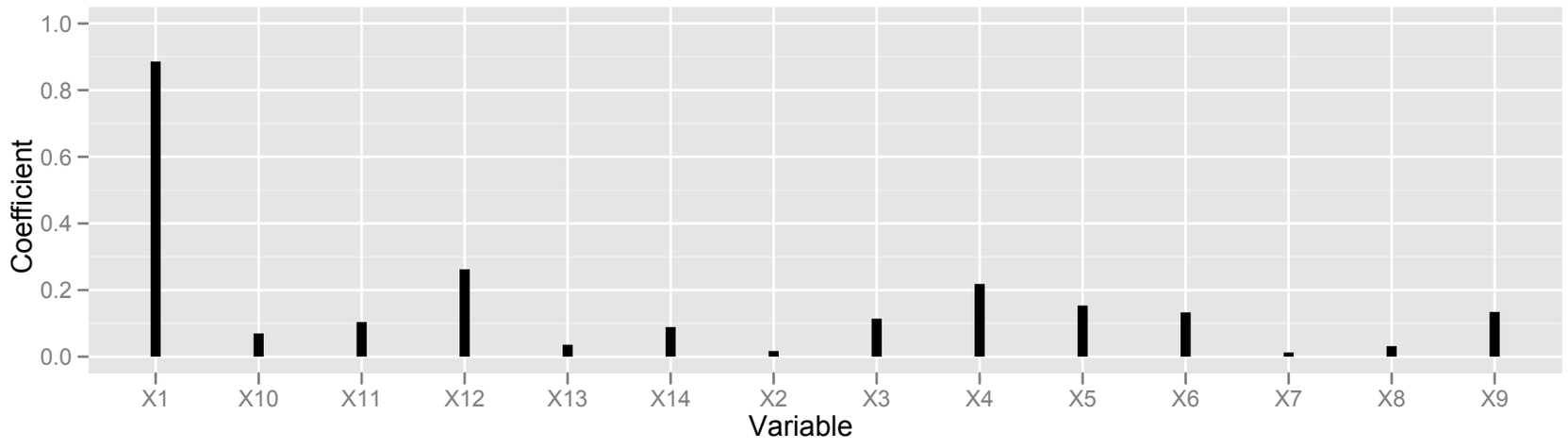
p=14

Good separation

error

	A	B
A	10	0
B	0	10



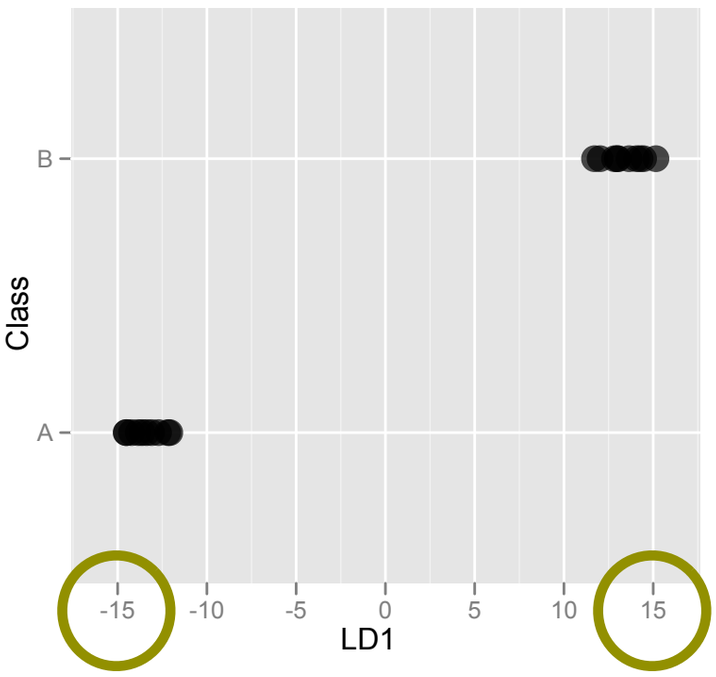


p=15

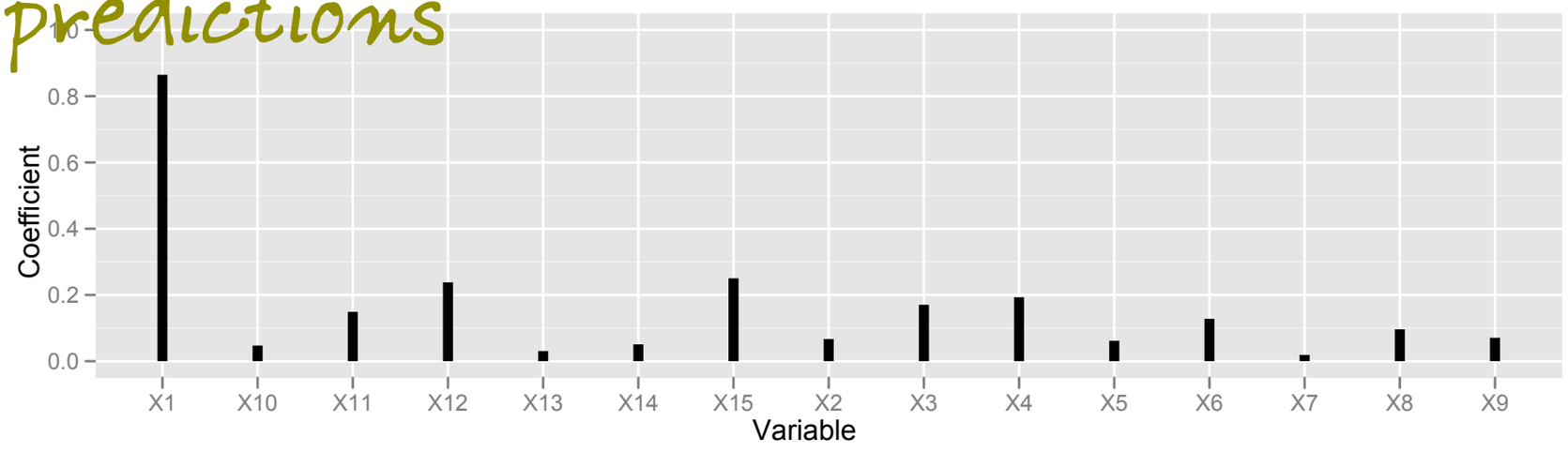
Data explodes out of the plot

error

	A	B
A	10	0
B	0	10



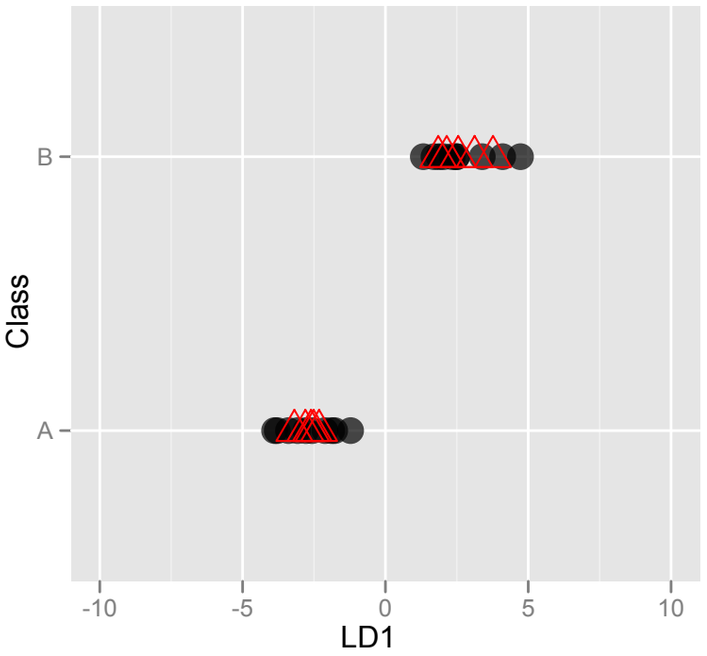
Still perfect predictions



Predict new data (p=2)

error	A	B
A	50	
B		5

Low error



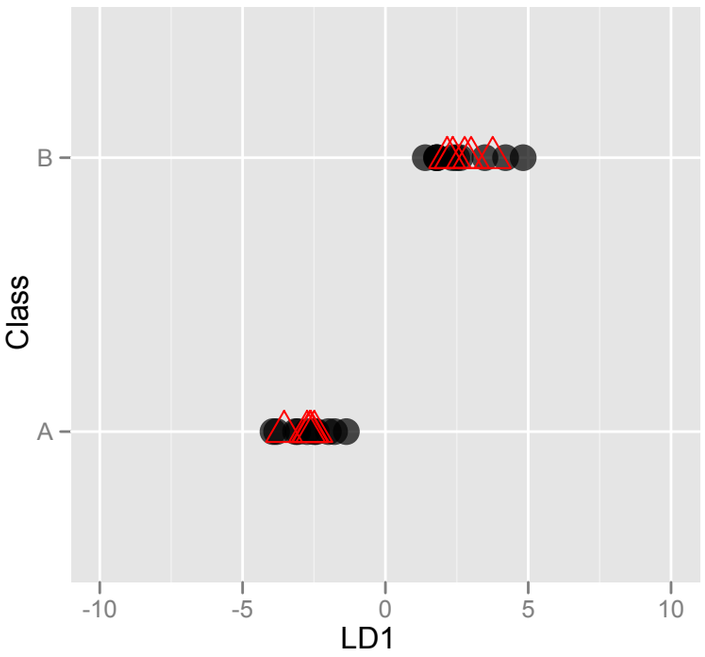
Test data matches training data

Predict new data (p=3)

error

	A	B
A	50	
B		50

Low error



Test data matches training data

Predict new data (p=4)

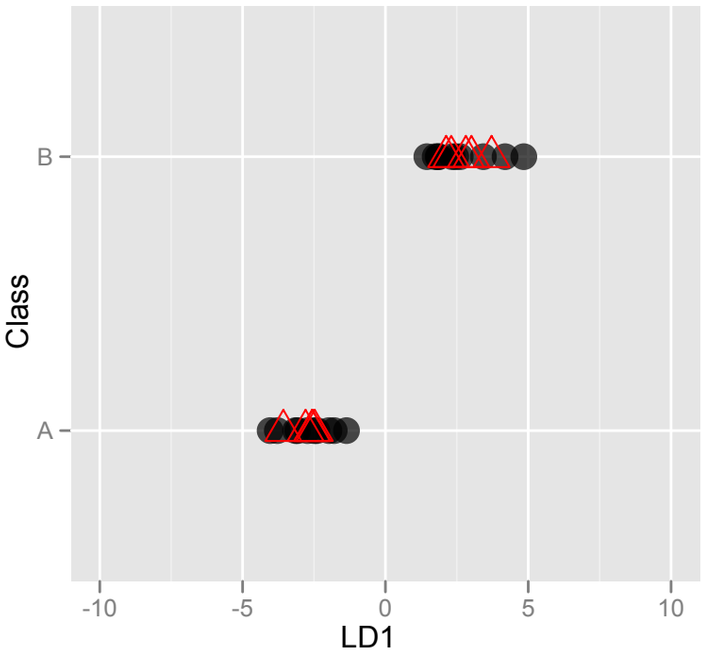
error

A B

A 5 0

B 0 5

Low error



Test data matches training data

Predict new data (p=5)

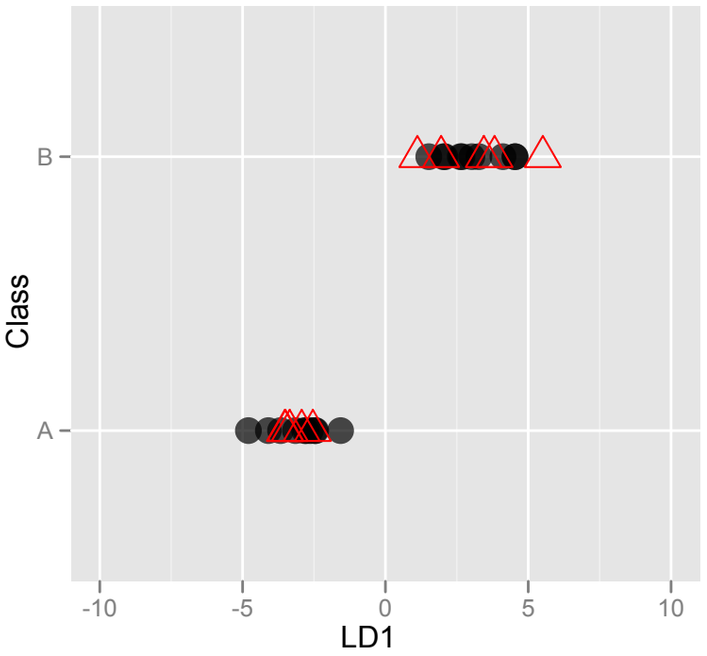
error

A B

A 5 0

B 0 5

Low error



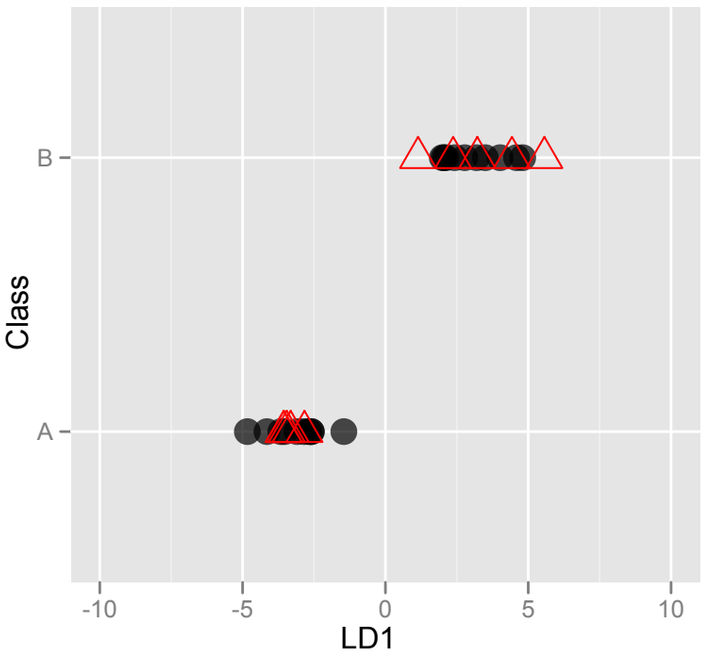
Test data not quite matching

Predict new data (p=6)

error

	A	B
A	5	0
B	0	5

Low error



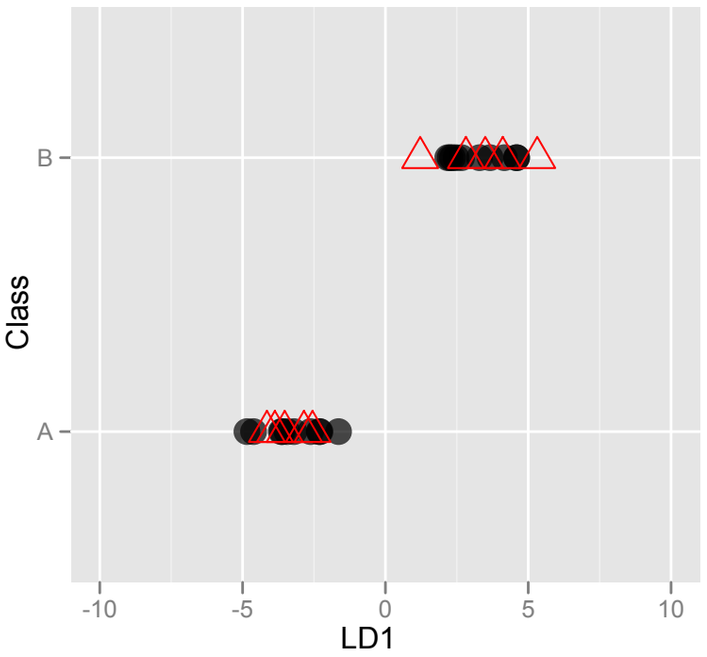
Test data not quite matching

Predict new data (p=7)

error

	A	B
A	5	0
B	0	5

Low error



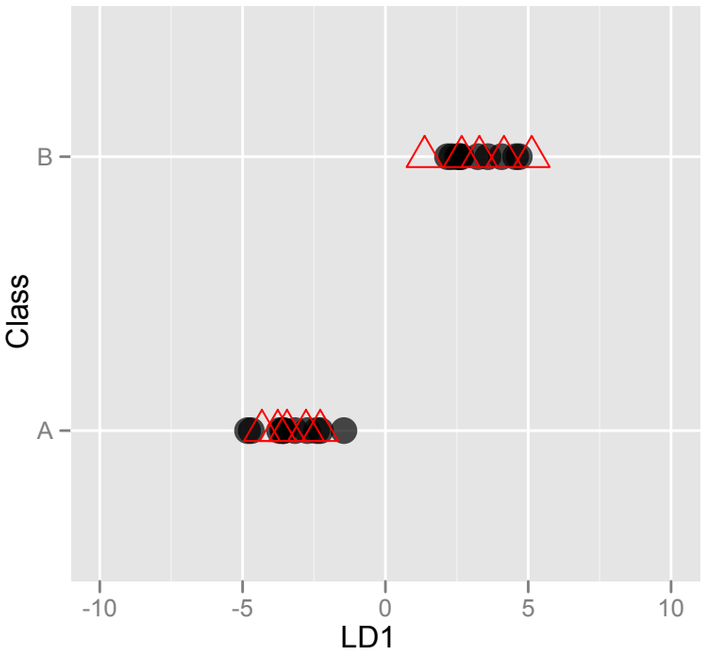
Test data not quite matching

Predict new data (p=8)

error

	A	B
A	50	
B		5

Low error



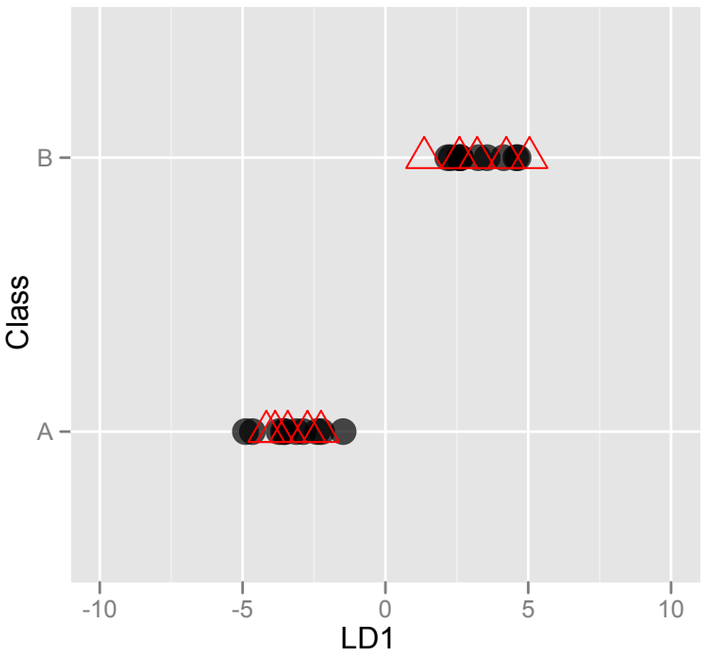
Test data not quite matching

Predict new data (p=9)

error

	A	B
A	50	
B		5

Low error



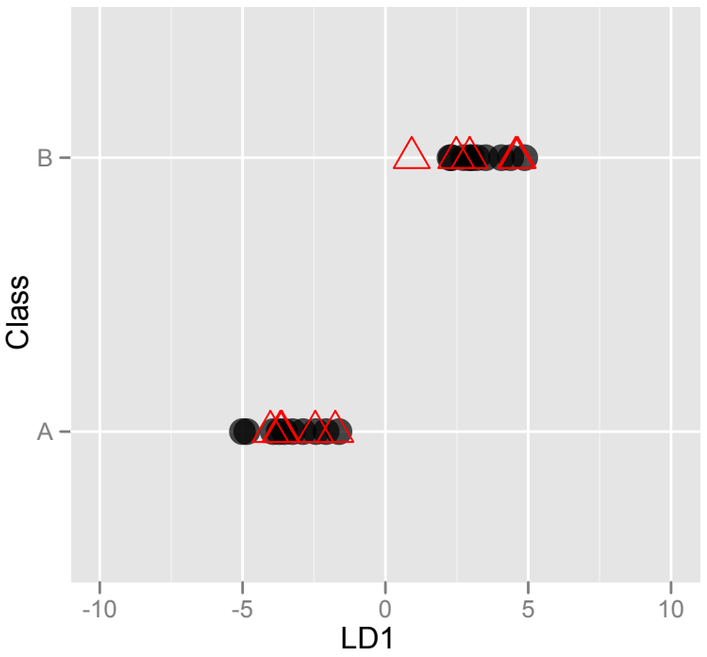
Test data not quite matching

Predict new data (p=10)

error

	A	B
A	5	0
B	0	5

Low error



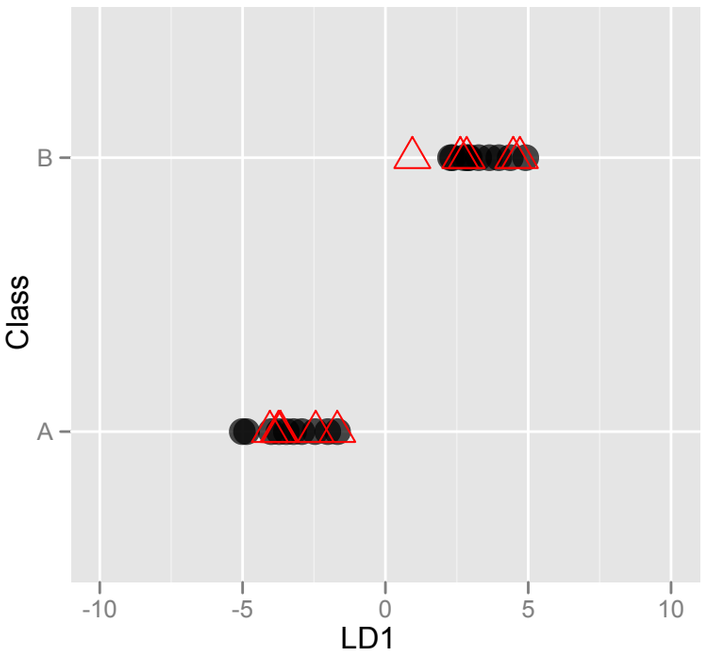
Test data not quite matching

Predict new data (p=11)

error

	A	B
A	50	
B		5

Low error



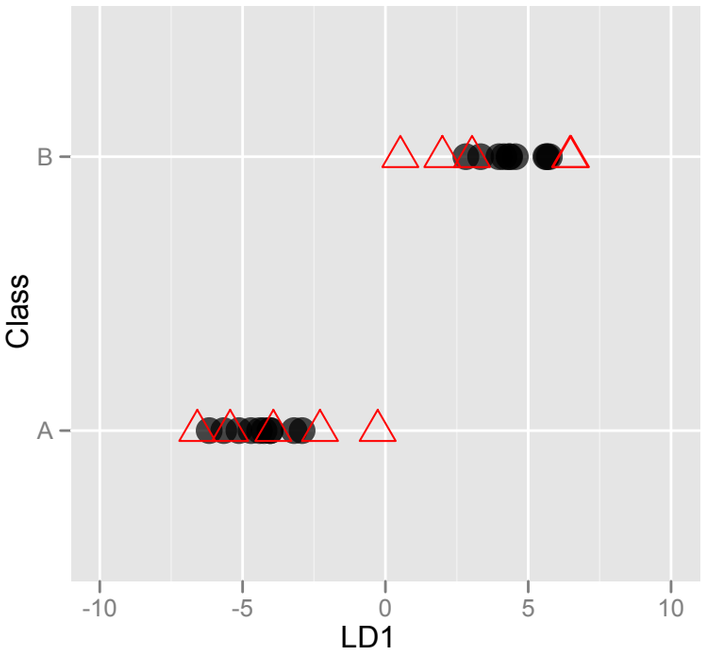
Test data not quite matching

Predict new data (p=12)

error

	A	B
A	5	0
B	0	5

Low error



Test data really not matching

Predict new data (p=13)

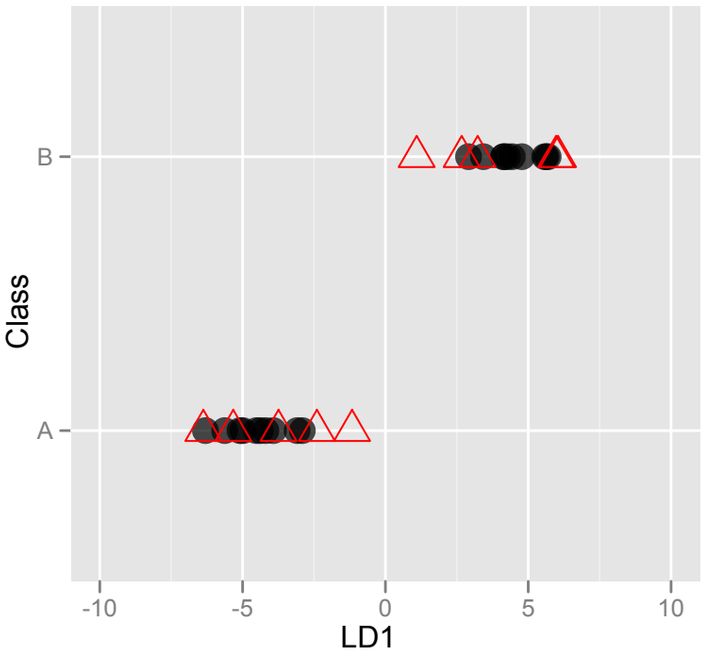
error

A B

A 5 0

B 0 5

Low error



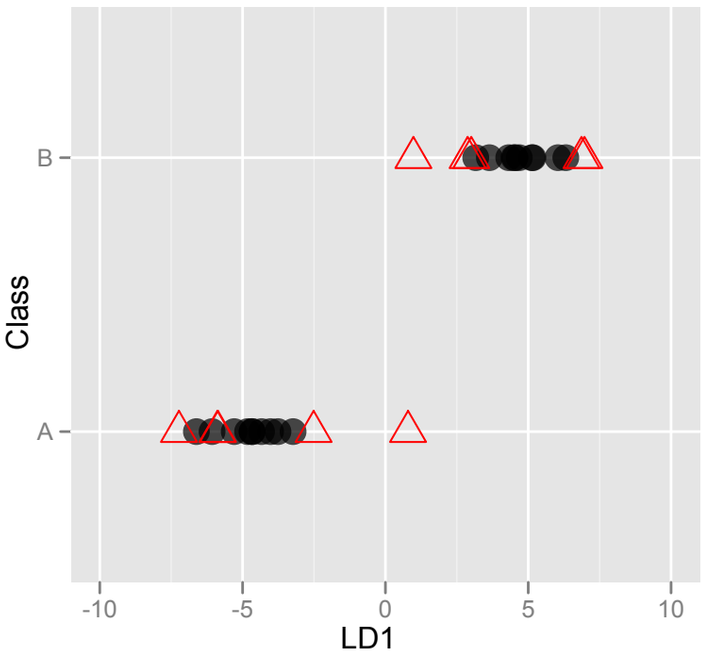
Test data really not matching

Predict new data (p=14)

error

	A	B
A	4	1
B	0	5

Error not 0



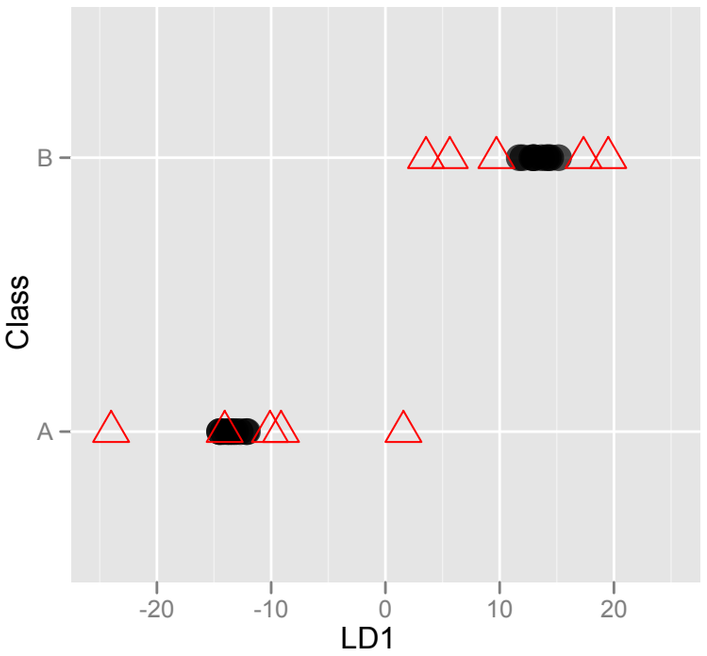
Test data really not matching

Predict new data (p=15)

error

	A	B
A	41	
B	0	5

Error not zero



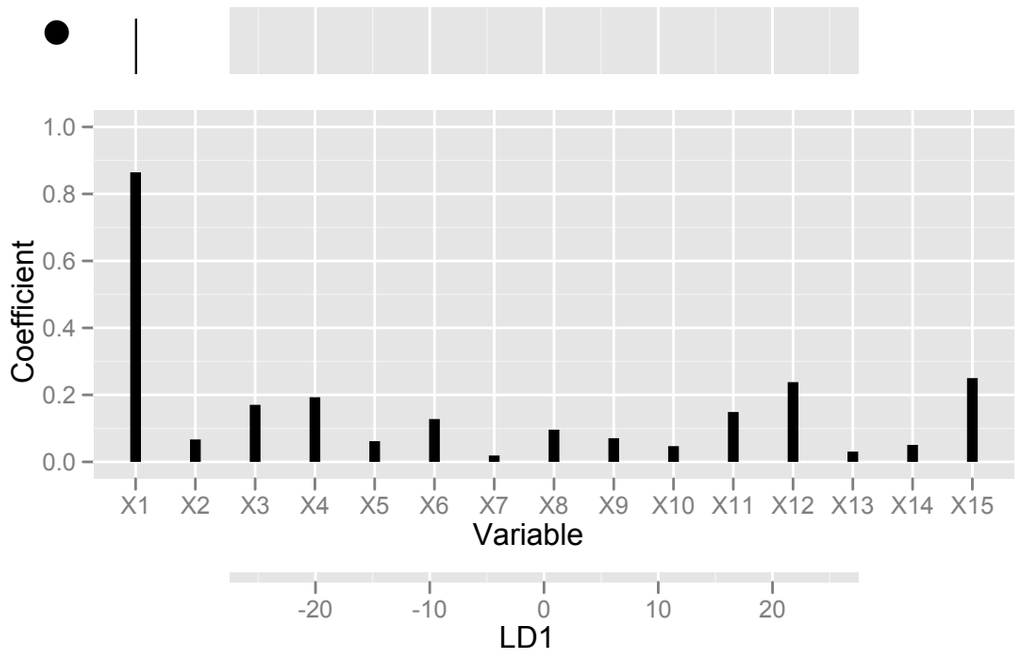
Test data really different from training data

PenalizedLDA package in R

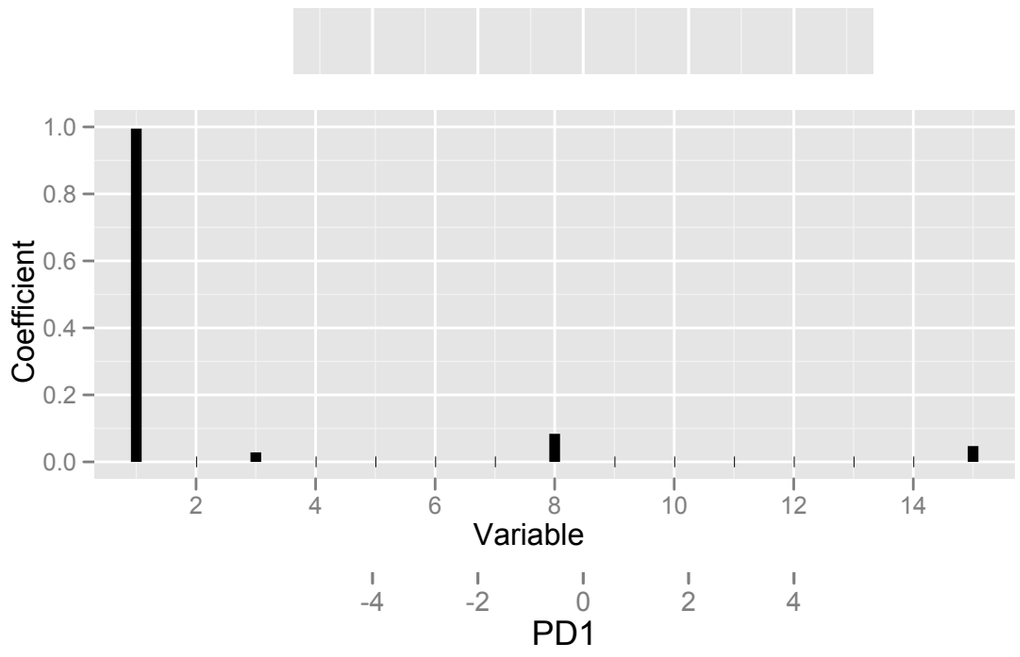
Penalized LDA

Maximize this $\mathbf{A}'((1-\lambda)\mathbf{S}+\lambda\mathbf{I})\mathbf{A}, \quad \lambda \in [0,1]$

LDA



PLDA



Perceptual principles

- Cleveland & McGill (1984)
- Data element to graphical element in rank order of accuracy in returning data value, is as follows:
  1. Position - common scale
  2. Position - nonaligned scale
  3. Length, direction, angle
  4. Area
  5. Volume, curvature
  6. Shading, color
- Results corroborated by Heer & Bostock (2010)

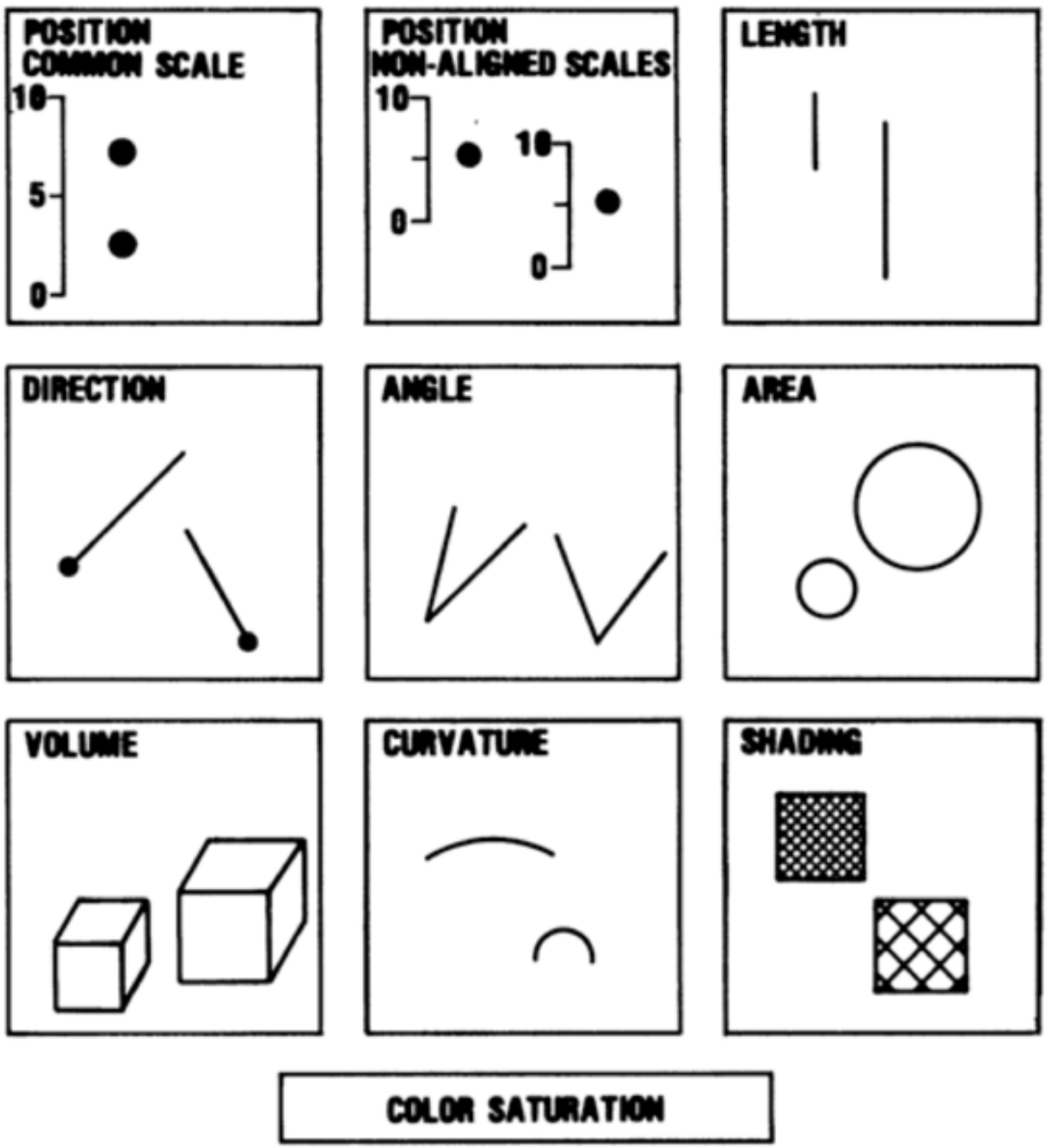


Figure 1. Elementary perceptual tasks.

Quantitative information should be mapped to position along a line, as first preference.

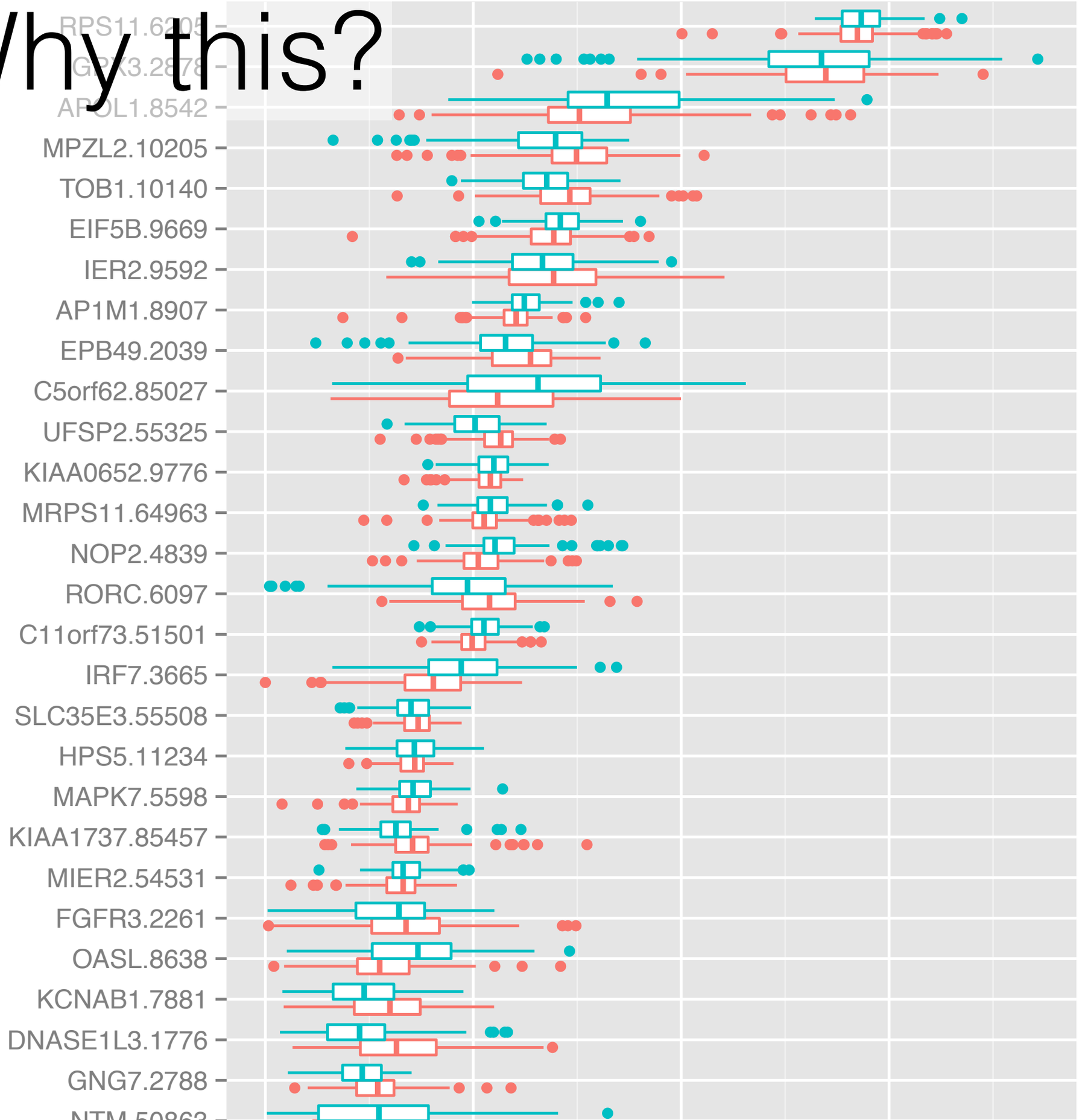
Proximity

- The closer the elements in a plot, the easier it is for the reader to compare them

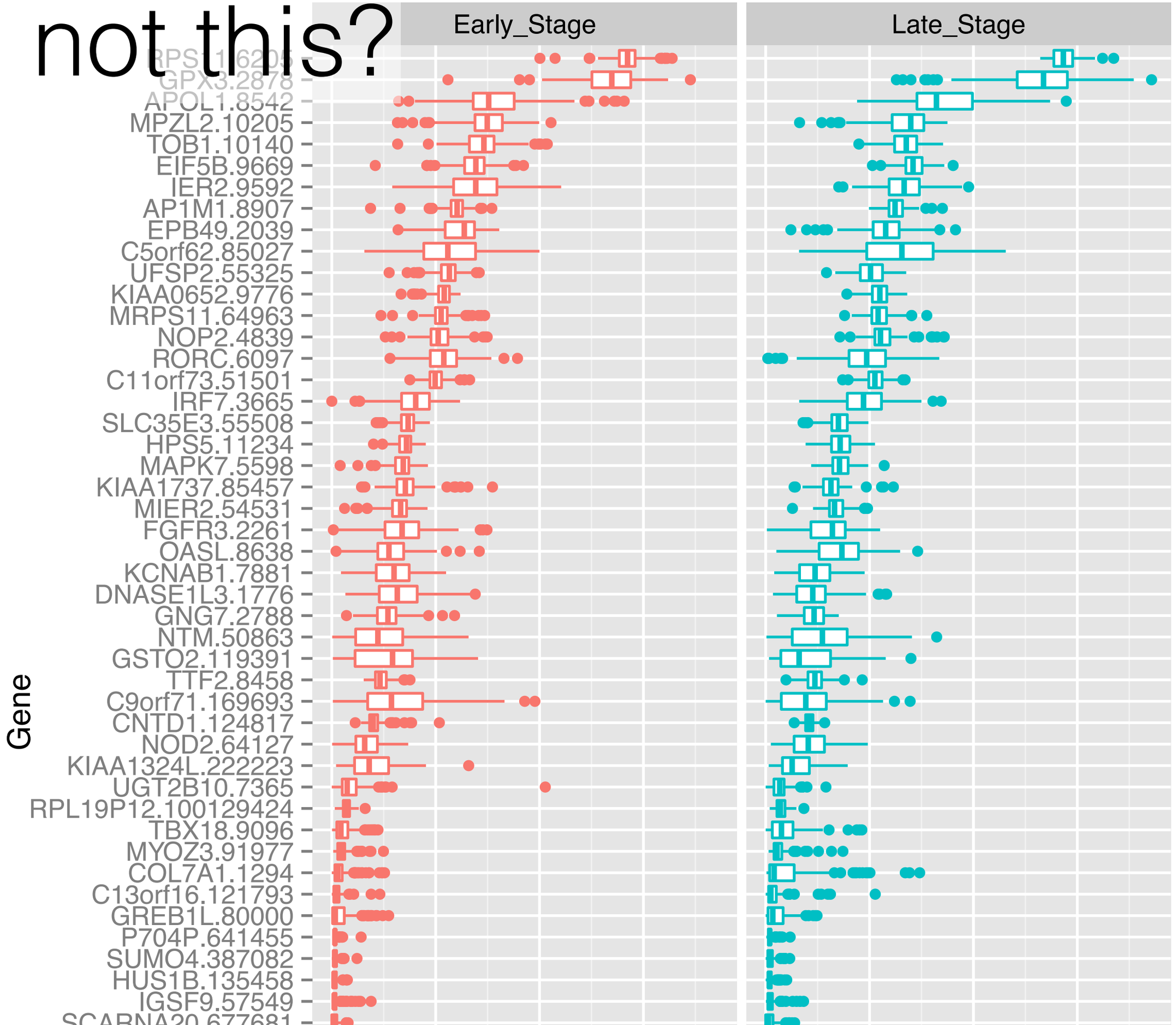
- Place elements for primary comparison close together.

How did we use proximity?

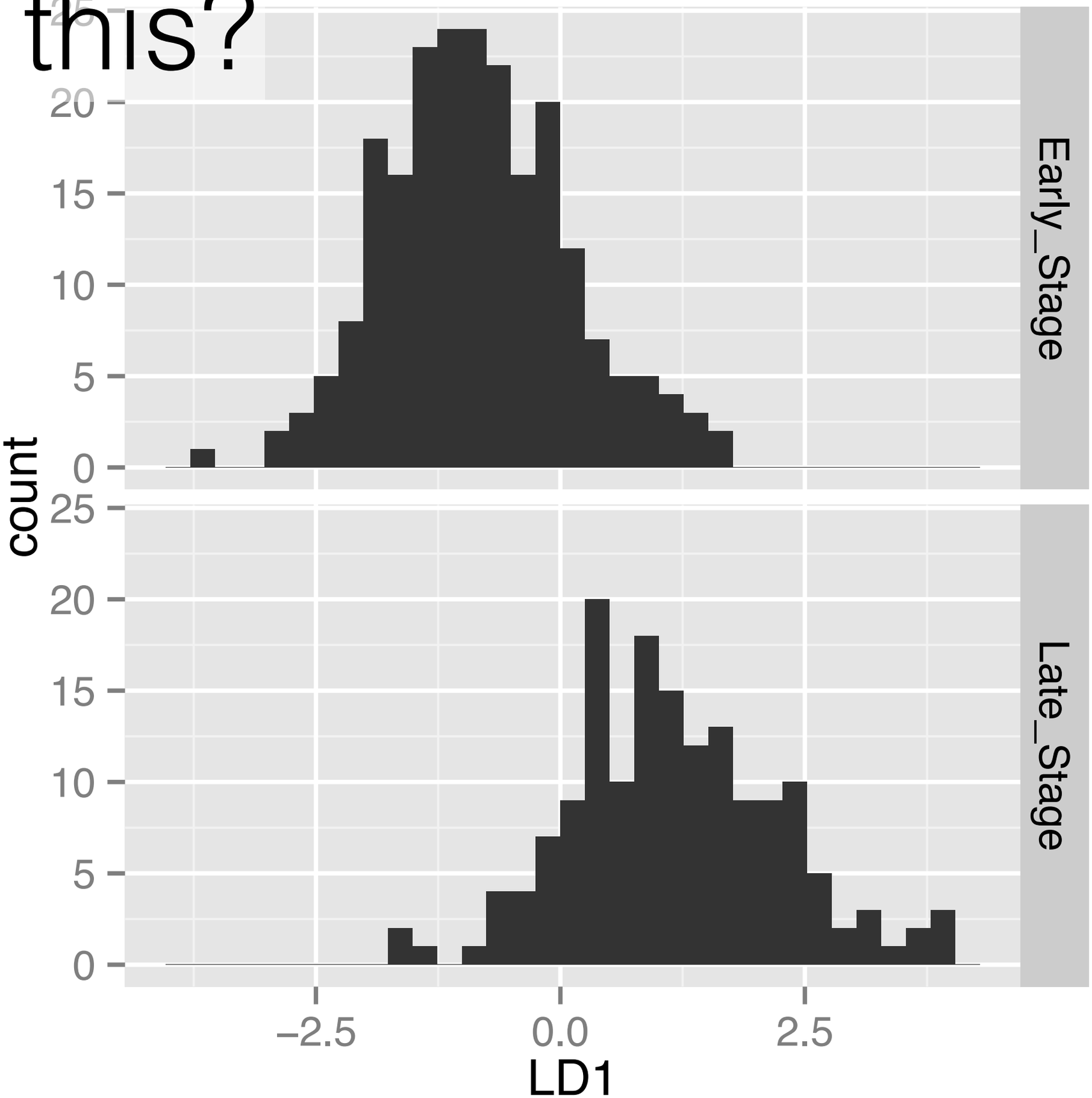
Why this?


RPS11.6205
GPX3.2878
APOL1.8542
MPZL2.10205
TOB1.10140
EIF5B.9669
IER2.9592
AP1M1.8907
EPB49.2039
C5orf62.85027
UFSP2.55325
KIAA0652.9776
MRPS11.64963
NOP2.4839
RORC.6097
C11orf73.51501
IRF7.3665
SLC35E3.55508
HPS5.11234
MAPK7.5598
KIAA1737.85457
MIER2.54531
FGFR3.2261
OASL.8638
KCNAB1.7881
DNASE1L3.1776
GNG7.2788

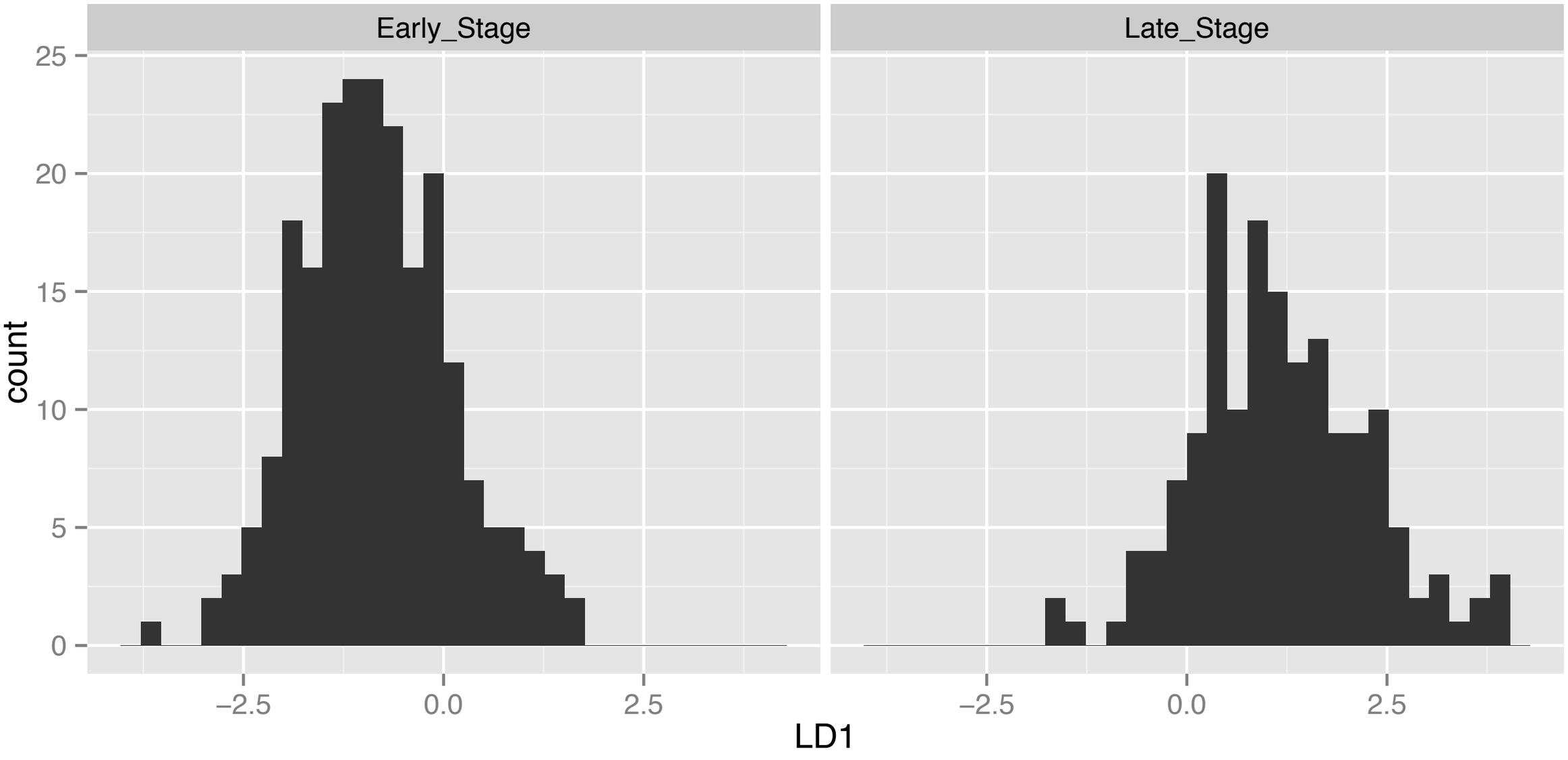
and not this?


Early_Stage
Late_Stage
Gene
RPS11.6205
GPX3.2878
APOL1.8542
MPZL2.10205
TOB1.10140
EIF5B.9669
IER2.9592
AP1M1.8907
EPB49.2039
C5orf62.85027
UFSP2.55325
KIAA0652.9776
MRPS11.64963
NOP2.4839
RORC.6097
C11orf73.51501
IRF7.3665
SLC35E3.55508
HPS5.11234
MAPK7.5598
KIAA1737.85457
MIER2.54531
FGFR3.2261
OASL.8638
KCNAB1.7881
DNASE1L3.1776
GNG7.2788
NTM.50863
GSTO2.119391
TTF2.8458
C9orf71.169693
CNTD1.124817
NOD2.64127
KIAA1324L.222223
UGT2B10.7365
RPL19P12.100129424
TBX18.9096
MYOZ3.91977
COL7A1.1294
C13orf16.121793
GREB1L.80000
P704P.641455
SUMO4.387082
HUS1B.135458
IGSF9.57549

Why this?

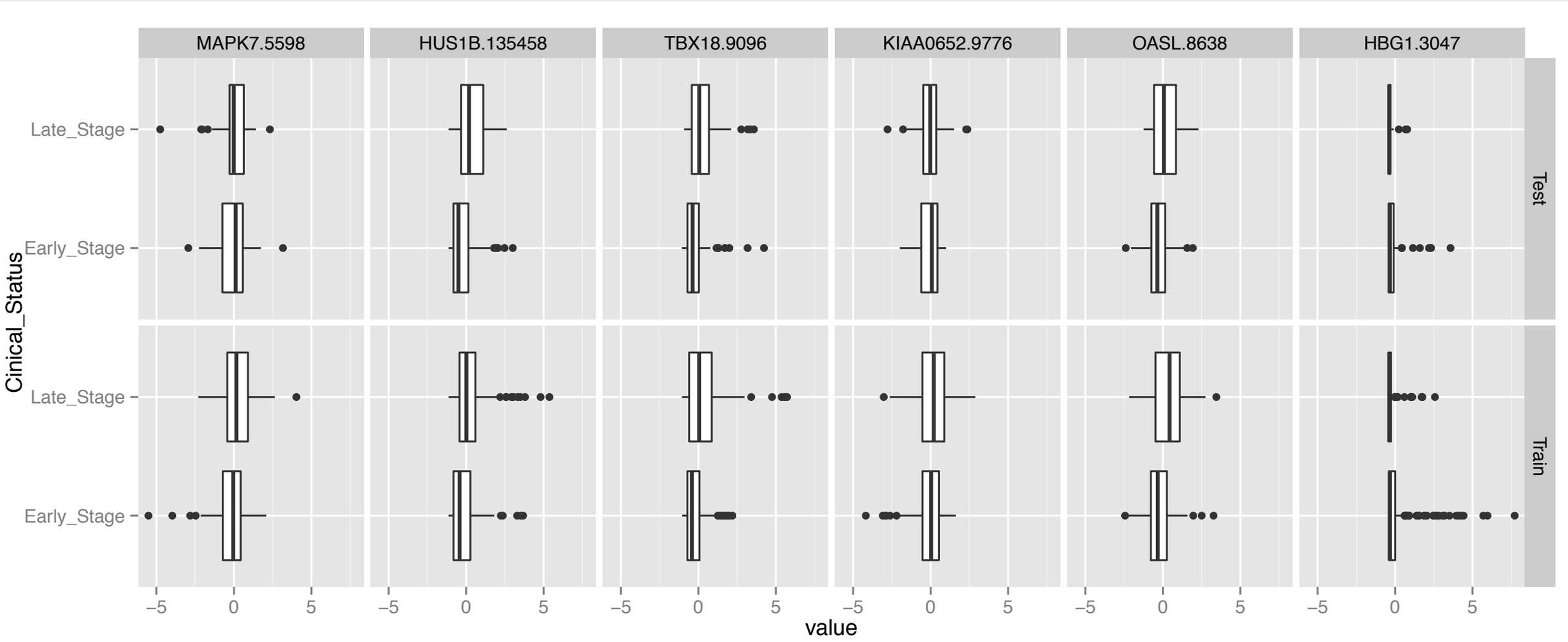


and not this?



Your Turn

- What other ways could we have arranged this plot?
- How would the primary comparison change?



Color

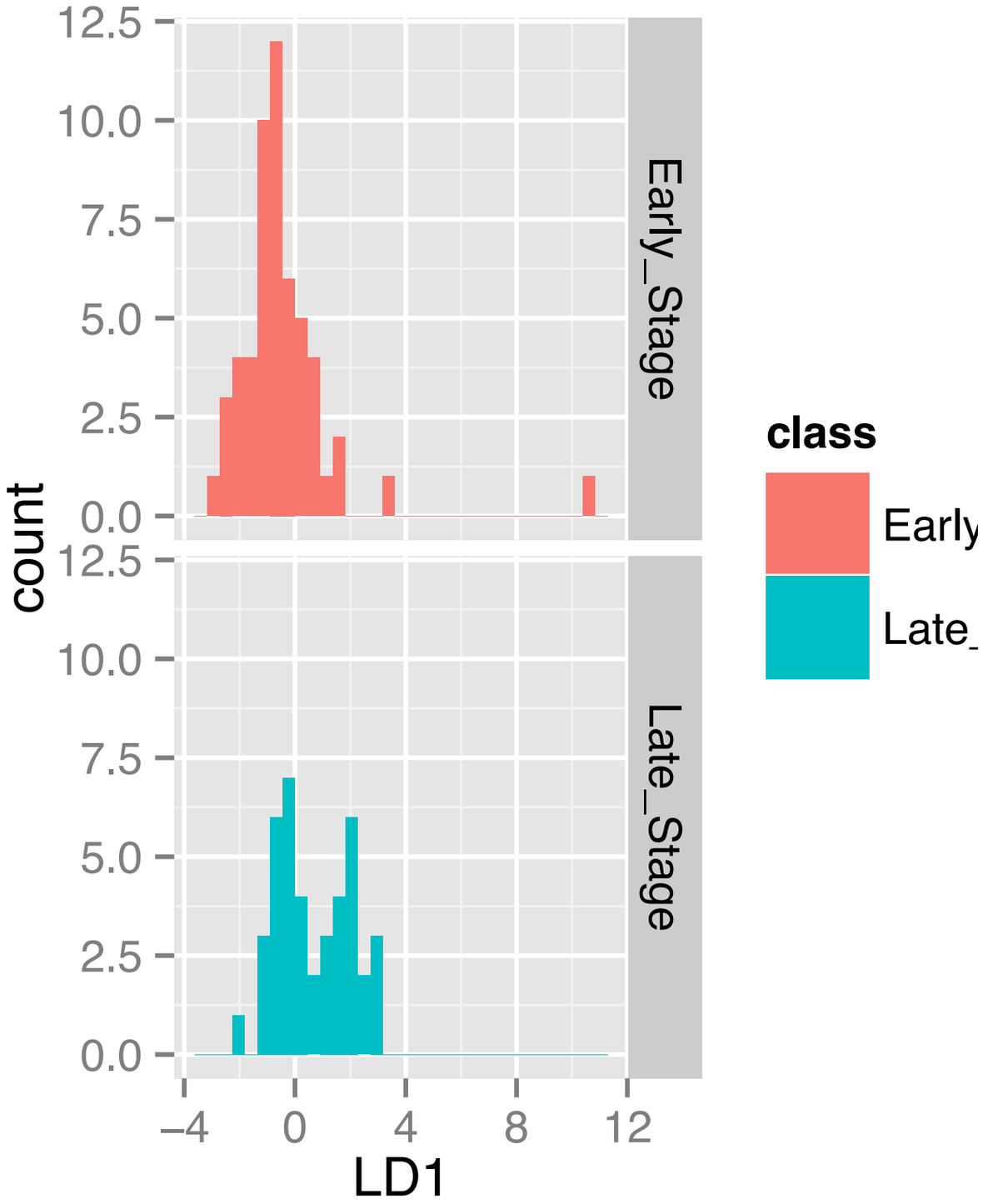
- Color is a pre-attentive graphical element, which means people see it before they realize they see it. If most elements are blue and one is red we will pick the red element out SO QUICKLY.
- Color is low on the scale of accurate reading.
- Map the appropriate information to the appropriate scale:
  - Use QUALITATIVE scales to display categorical data, small number of levels,
  - SEQUENTIAL scales are used to represent a gradient of quantitative information, e.g. 0-100
  - DIVERGING scales are used to represent negative to positive quantitative information.

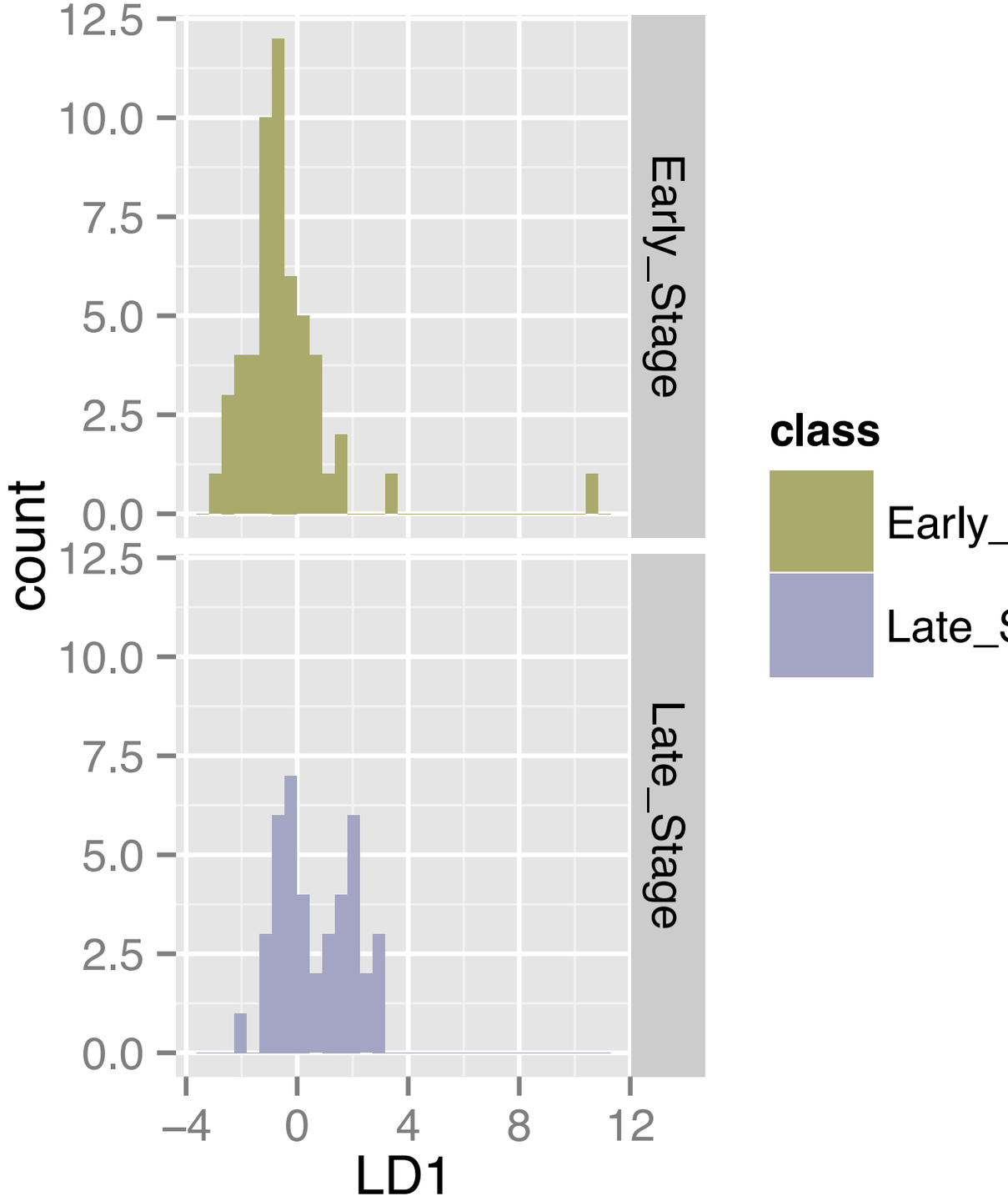
Color blindness

- Color choice can affect whether all your audience can see what you see.

- Use the package ‘dichromat’, to see what your plot looks like to a red-green color blind eye.

```
> library(dichromat)
> library(scales)
> clrs <- hue_pal()(2)
> qplot(LD1, data=ts.df, geom="histogram", facets=class~.,
   fill=class) + scale_fill_manual(values=clrs)
> clrs <- dichromat(hue_pal()(2))
> qplot(LD1, data=ts.df, geom="histogram", facets=class~.,
   fill=class) + scale_fill_manual(values=clrs)
```


count
12.5
10.0
7.5
5.0
2.5
0.0
Early_Stage
Late_Stage
class
Early
Late_
–4
0
4
8
12
LD1


count
12.5
10.0
7.5
5.0
2.5
0.0
Early_Stage
Late_Stage
class
Early_
Late_S
–4
0
4
8
12
LD1

Show the data

Genomic plots

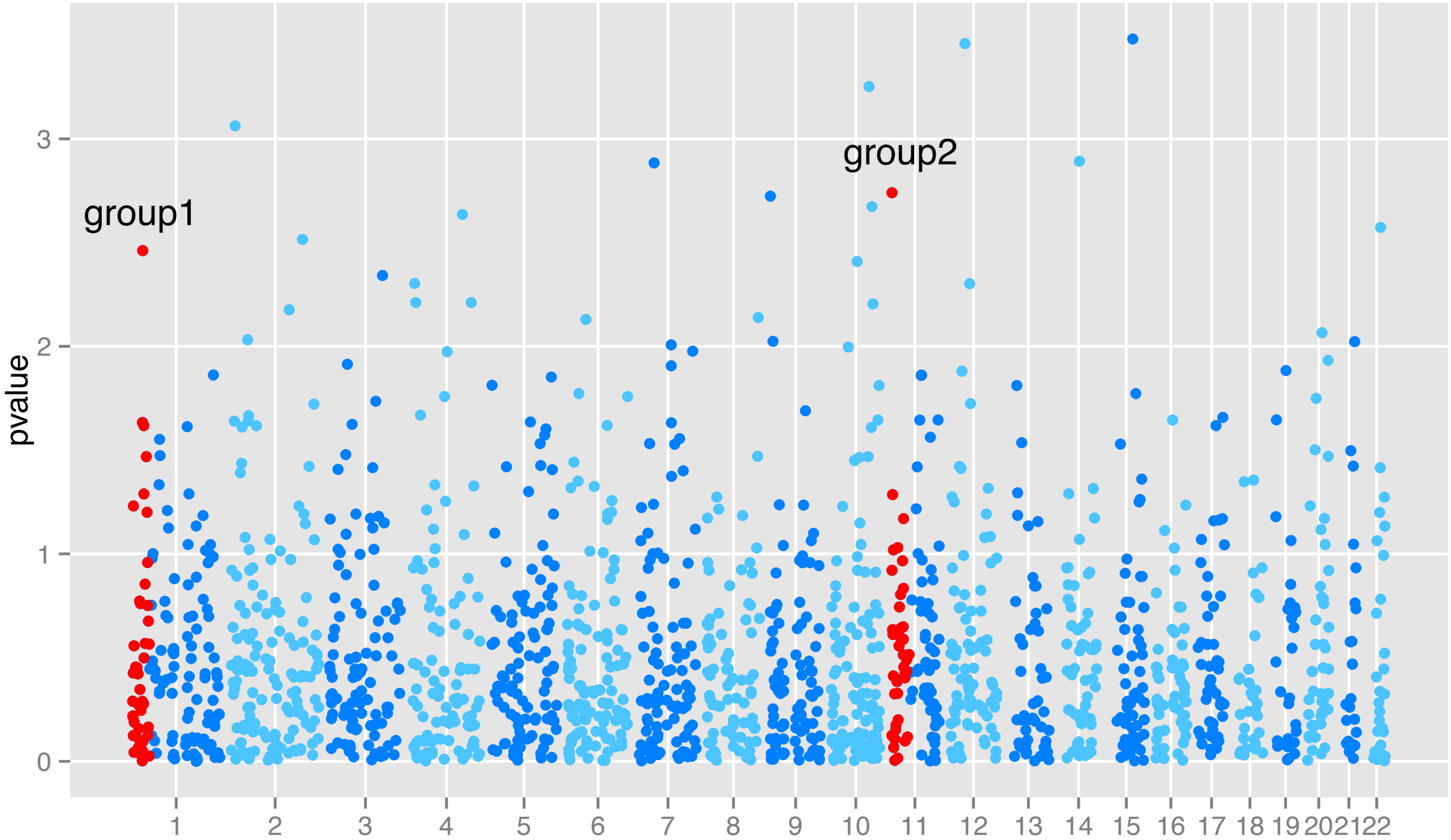
- The package ‘ggbio’ extends ggplot2 for making genomic plots

- The plots provide detailed views of genomic regions, summary views of sequence alignments and splicing patterns, and genome-wide overviews with karyogram, circular and grand linear layouts.

- Yin, Cook, Lawrence (2012) ggbio: an R package for extending the grammar of graphics for genomic data, Genome Biology 13:R77 http://genomebiology.com/content/13/8/R77

Manhattan plots

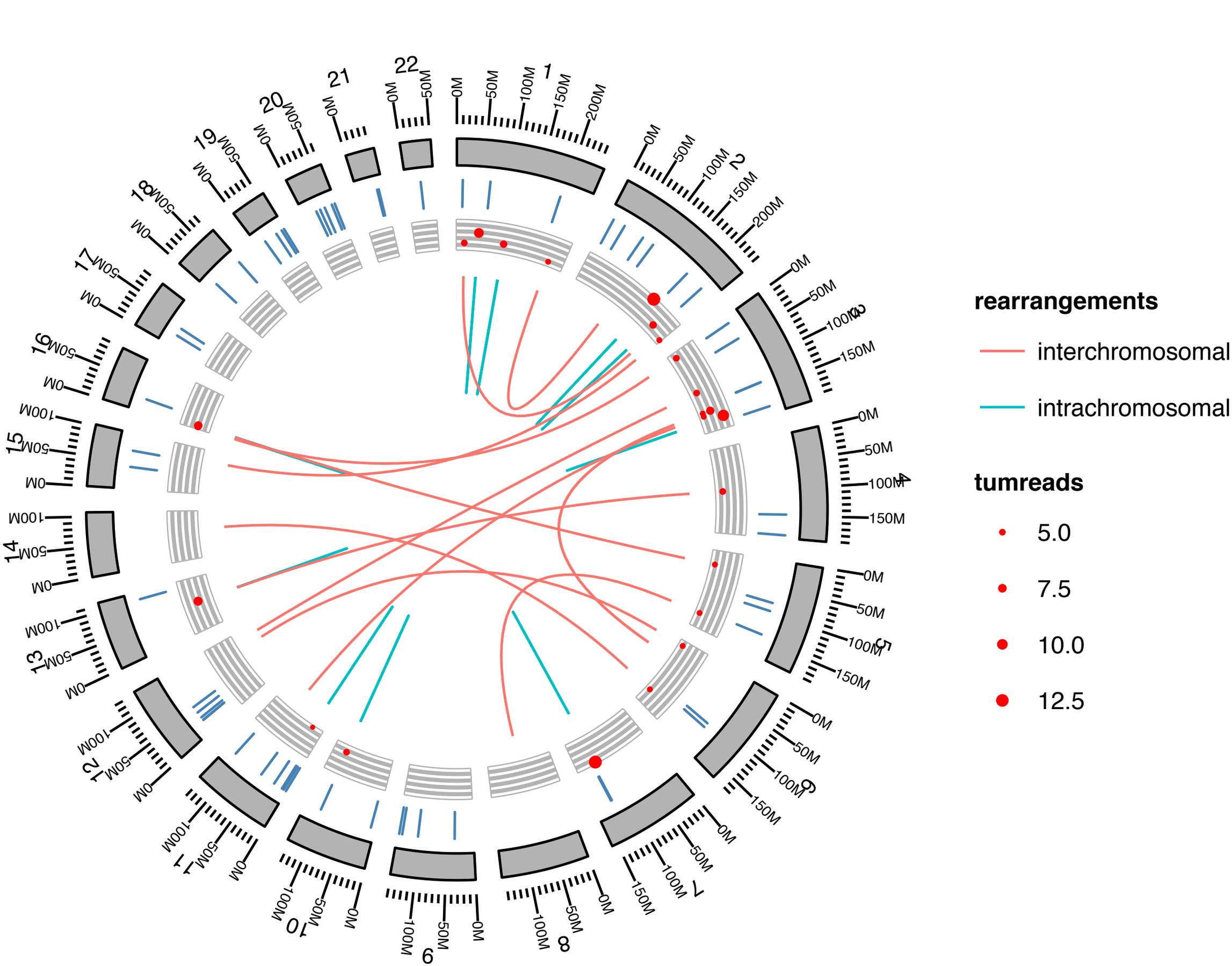
```
> snp <- read.table(system.file("extdata", "plink.assoc.sub.txt",
    package = "biovizBase"), header = TRUE)
> gr.snp <- transformDfToGr(snp, seqnames = "CHR", start = "BP",
    width = 1)
> head(gr.snp)
GRanges object with 6 ranges and 10 metadata columns:
    seqnames                ranges strand |       CHR         SNP        BP        A1
       <Rle>             <IRanges>  <Rle> | <integer>    <factor> <integer>  <factor>
  [1]        4 [ 10794096,  10794099]      * |         4   rs9291494  10794096         G
> gr.snp <- keepSeqlevels(gr.snp, as.character(1:22))
> data(ideoCyto, package = "biovizBase")
> seqlengths(gr.snp) <- as.numeric(seqlengths(ideoCyto$hg18)[1:22])
> gr.snp <- gr.snp[!is.na(gr.snp$P)]
> values(gr.snp)$pvalue <- -log10(values(gr.snp)$P)
> gro <- GRanges(c("1", "11"), IRanges(c(100, 2e6), width = 5e7))
> names(gro) <- c("group1", "group2")
> plotGrandLinear(gr.snp, aes(y = pvalue), highlight.gr = gro)
```


group1
group2
pvalue
3
2
1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

Circular layout

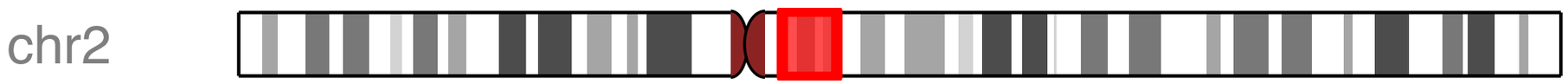
```
> data("CRC", package = "biovizBase")
> head(hg19sub)
GRanges object with 6 ranges and 0 metadata columns:
      seqnames          ranges strand
         <Rle>       <IRanges>  <Rle>
  [1]        1 [1, 249250621]      *
  [2]        2 [1, 243199373]      *
```

```
> p <- ggbio() +
    circle(gr.crc1, geom = "link", linked.to = "to.gr",
      aes(color = rearrangements)) +
    circle(gr.crc1, geom = "point", aes(y = score, size = tumreads),
      color = "red", grid = TRUE) + scale_size(range = c(1, 2.5)) +
    circle(mut.gr, geom = "rect", color = "steelblue") +
    circle(hg19sub, geom = "ideo", fill = "gray70") +
    circle(hg19sub, geom = "scale", size = 2) +
    circle(hg19sub, geom = "text", aes(label = seqnames),
      vjust = 0, size = 3)
```


rearrangements
interchromosomal
intrachromosomal
tumreads
5.0
7.5
10.0
12.5

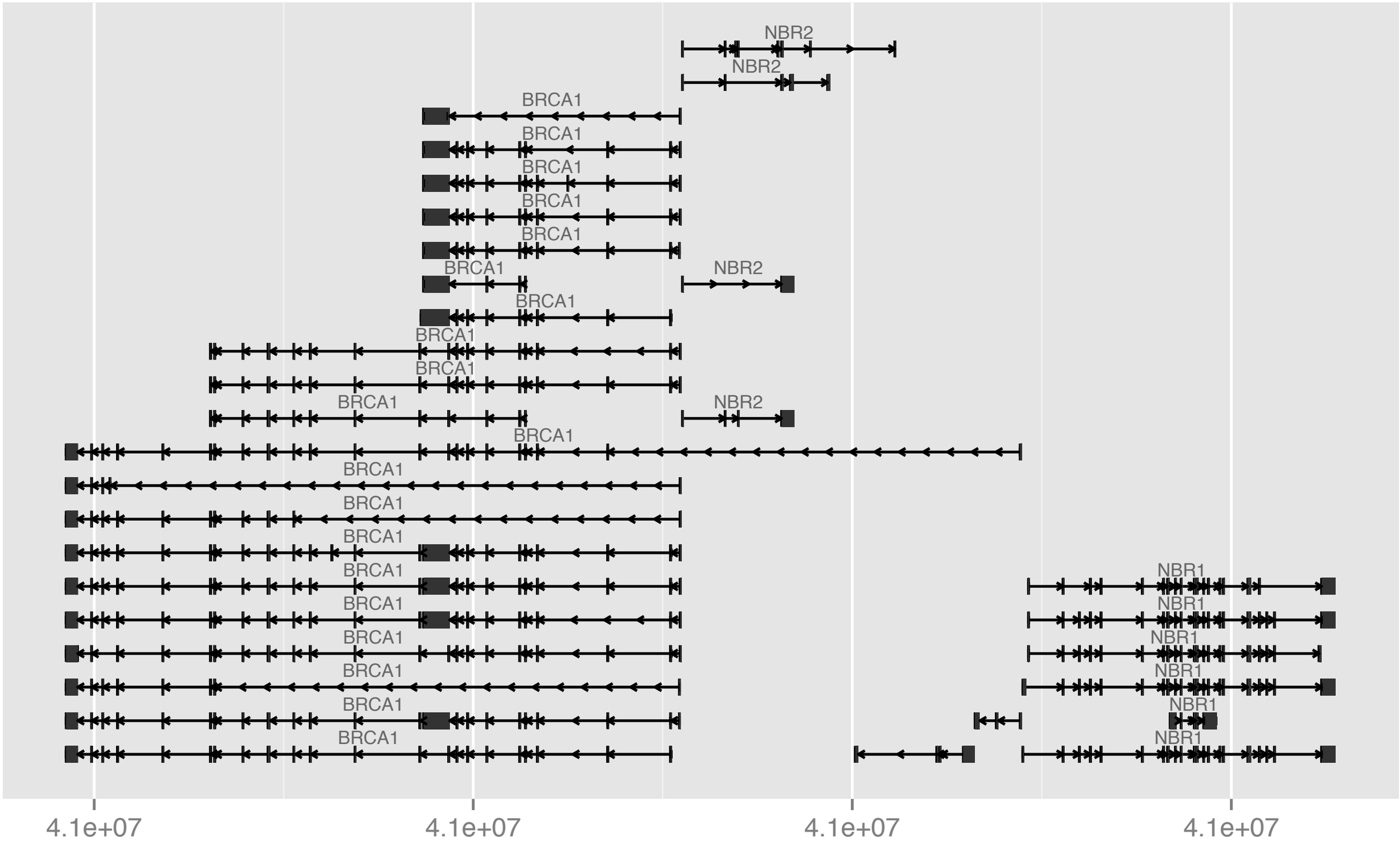
Ideogram

```
> p.ideo <- Ideogram(genome = "hg19")
> library(GenomicRanges)
> ## special highlights instead of zoomin!
> p.ideo + xlim(GRanges("chr2", IRanges(1e8, 1e8+10000000)))
```



Gene Model

```
> library(Homo.sapiens)
> class(Homo.sapiens)
[1] "OrganismDb"
attr(,"package")
[1] "OrganismDbi"
> data(genesymbol, package = "biovizBase")
> wh <- genesymbol[c("BRCA1", "NBR1")]
> wh <- range(wh, ignore.strand = TRUE)
> p.txdb <- autoplot(Homo.sapiens, which  = wh)
Parsing transcripts...
Parsing exons…
> p.txdb
```


NBR2
NBR2
BRCA1
BRCA1
BRCA1
BRCA1
BRCA1
BRCA1
NBR2
BRCA1
BRCA1
BRCA1
NBR2
BRCA1
BRCA1
BRCA1
BRCA1
BRCA1
BRCA1
BRCA1
BRCA1
NBR1
NBR1
NBR1
NBR1
NBR1
NBR1
4.1e+07
4.1e+07
4.1e+07
4.1e+07