STAT 545A
Class meeting #10
Wednesday, October 10, 2012

Dr. Jennifer (Jenny) Bryan

Department of Statistics and Michael Smith Laboratories

Review of last class

Develop an R coding style. Keep your projects organized.

Use names. Avoid Magic Numbers. Break tasks and scripts in small pieces. Avoid hand-hacking, e.g. by writing functions.

Take advantage of .Rprofile and .Renviron to make persistent changes to your R life.

Manage your R installation so that upgrading once or twice a year is relatively painless.

Source is real. Other things ... not so much.

Make sure you can reproduce your analytical results and your figures from the input data without breaking much of a sweat.

Review of last class

Keep your eyes/ears/mind open about professional tools for reproducible research, collaboration and open science, and version control. You will probably need to address this as your training and career goes on.

Look at Sweave and knitr for creating dynamic, integrated documents of a data analysis. JB favors knitr, based on R Markdown, since markdown comes up in so many other places.

Consider git (JB's choice), mercurial, and Subversion (SVN) for version control.

If you choose git, then github is a great way to share your work with others and/or to collaborate.

Review of last class

(Semi-?) automate getting non-figure stuff out of R and ready for inclusion into, e.g. manuscripts, webpages.

Use write.table for spreadsheet-y stuff. dput / save / sink useful in special cases. Use packages like xtable / Hmisc / R2HTML for writing HTML and LaTeX tables.

Get started on your final project!

Do not underestimate the time and effort required to capture and tame a "wild" dataset (versus those found in captivity, i.e. data analyzed ad nauseum in textbooks, in previous classes you took, in R documentation, etc.). There will be weird stuff you must [1] detect and [2] fix.

How to ask a question to maximize chance of getting an answer.

The R Inferno

Patrick Burns[1]

30th April 2011

Read about the 9th circle of R hell in The R Inferno.

Circle 9

Homework reading

Unhelpfully Seeking Help

Here live the thieves, guarded by the centaur Cacus. The inhabitants are bitten by lizards and snakes.

There's a special place for those who—not being content with one of the 8 Circles we've already visited—feel compelled to drag the rest of us into hell.

The road to writing a mail message should include at least the following stops:

9.1 Read the appropriate documentation.

"RTFM" in the jargon. There is a large amount of documentation about R, both official and contributed, and in various formats. A large amount of documentation means that it is often nontrivial to find what you are looking for—especially when frustration is setting in and blood pressure is rising.

Breathe.

There are various searches that you can do. R functions for searching include `help.search`, `RSiteSearch` and `apropos`.

If you are looking for particular functionality, then check the Task Views (found on the left-side menu of CRAN).

If you have an error, then look in rather than out—debug the problem. One way of debugging is to set the `error` option, and then use the `debugger` function:

```
options(error=dump.frames)
# command that causes the error
debugger()
```

The `debugger` function then provides a menu of the stack of functions that have been called at the point of the error. You can inspect the state of the objects inside these functions, and hopefully understand what the problem is.

The R Inferno

Patrick Burns[1]

30th April 2011

Circle 9

Unhelpfully Seeking Help

Here live the thieves, guarded by the centaur Cacus. The inhabitants are bitten by lizards and snakes.

There's a special place for those who—not being content with one of the 8 Circles we've already visited—feel compelled to drag the rest of us into hell.

"If someone has the wit and knowledge to answer your question, they probably have other things they would like to do. Making your message clear, concise and user-friendly gives you the best hope of at least one of those strangers diverting their attention away from their life towards your problem."

How to ask a question to maximize chance of getting an answer.

Homework reading

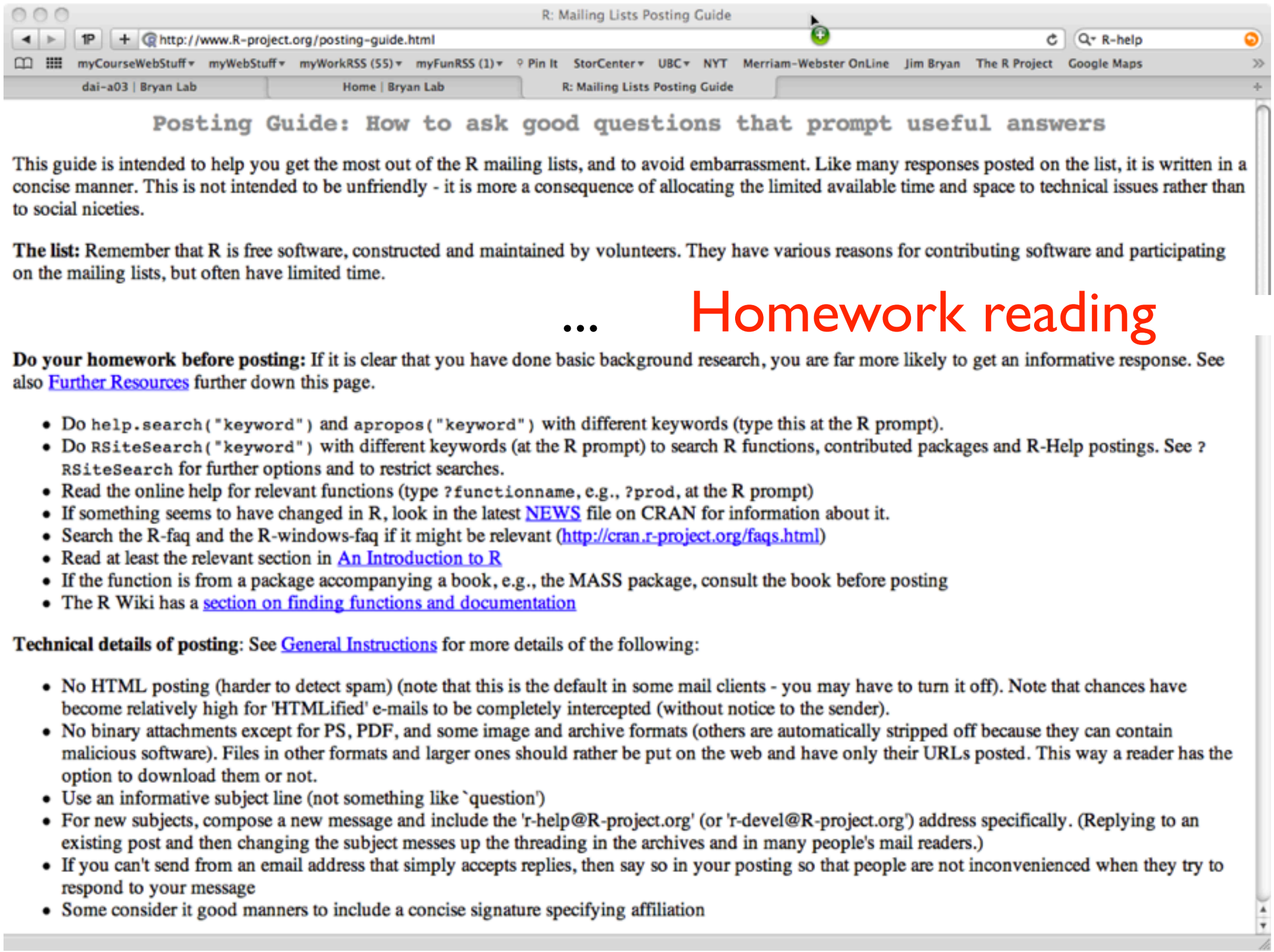
"How To Ask Questions The Smart Way"
by Eric Raymond and Rick Moen

It's OK to be ignorant; it's not OK to play stupid.

So, while it isn't necessary to already be technically competent to get attention from us, it is necessary to **demonstrate the kind of attitude that leads to competence: alert, thoughtful, observant, willing to be an active partner in developing a solution**.

The best way to get a rapid and responsive answer is to ask it like a person with smarts, confidence, and clues who just happens to need help on one particular problem.

R-help (you probably shouldn't be posting here!) has a good posting guide



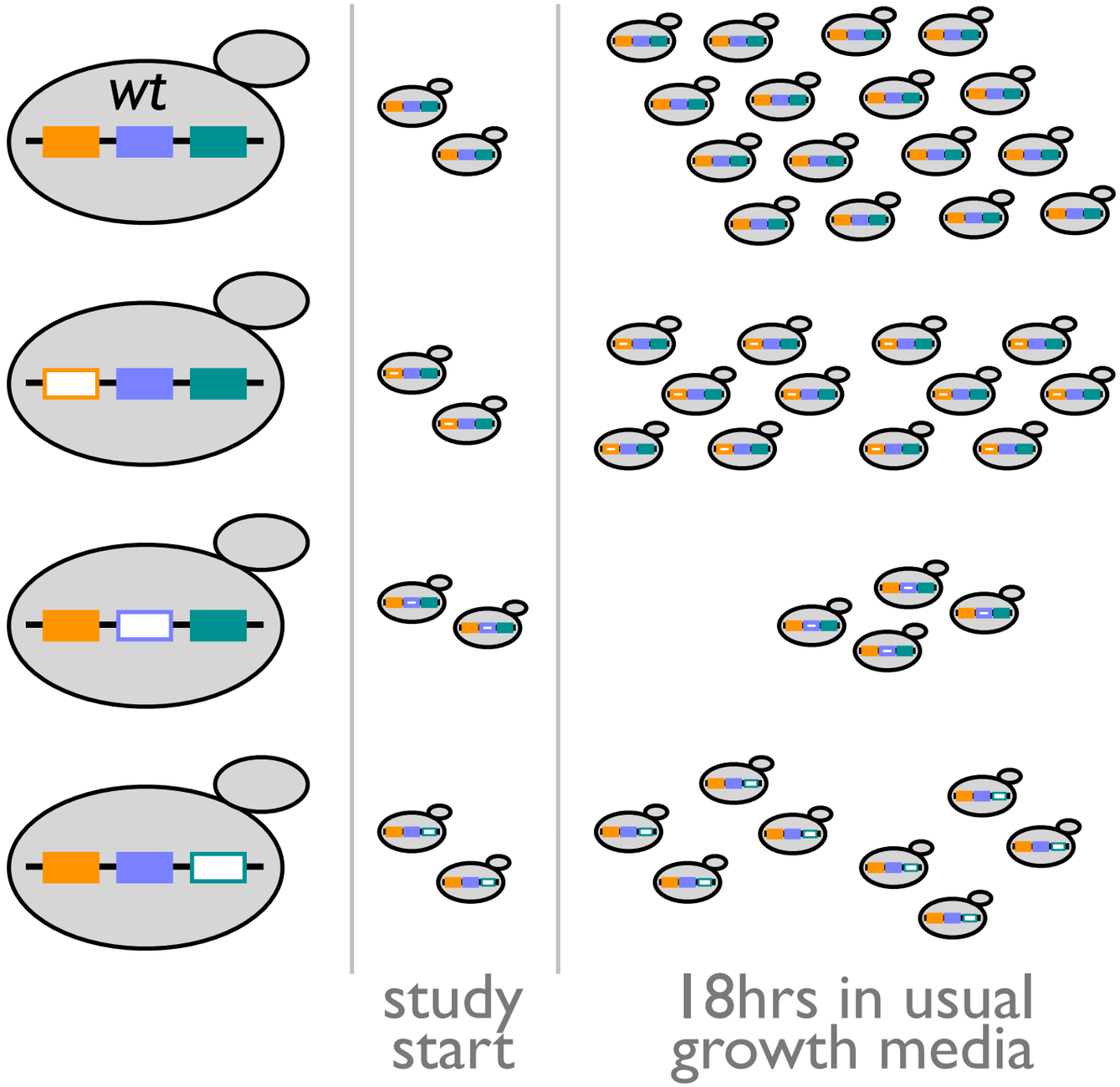
Where we go from here

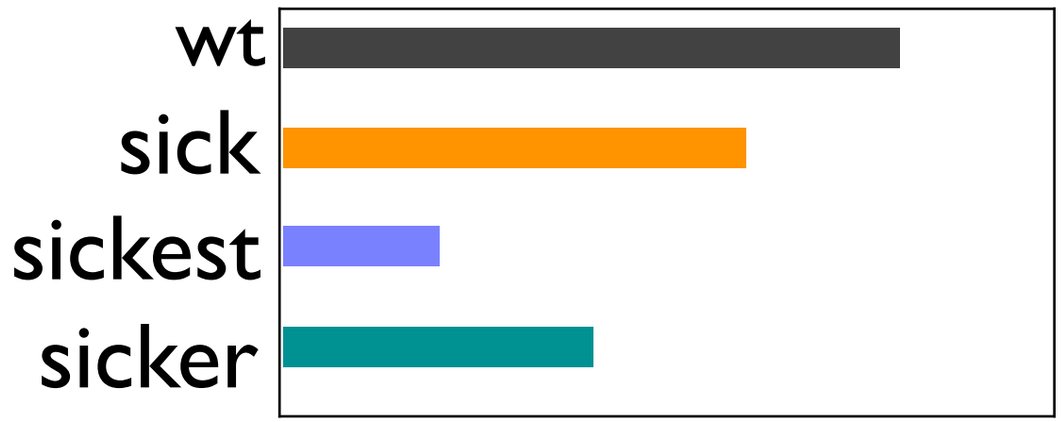
Today: Two group tests. "Scaling up": different tests, many two group comparisons. Using figures to convey bulk statistical results instead of big tables of numbers. Use two group testing problem as way to introduce the bootstrap approach.

Next Monday: Continuing to develop and illustrate the bootstrap. Robust regression.

Next Wednesday: Smooth regression and cross validation.

Study of fitness of yeast deletion mutants





Rationale for growth studies yeast deletion mutants

- Analogy: flipping circuit breakers in a house to determine which lights and outlets are controlled by each circuit
- If the deletion mutant for gene *g* is defective at some biological activity, that suggests that gene *g* contributes to that activity.
- Growth studies are the 'entry-level' study. In real life, we often measure more complicated phenotypes and subject the mutant to additional challenges, e.g. treatment with drugs or deletion/mutation of additional genes. Also, this type of data is often integrated with other types of studies.

Y = a quantitative measure of growth

e.g. growth rate or # cells at study end

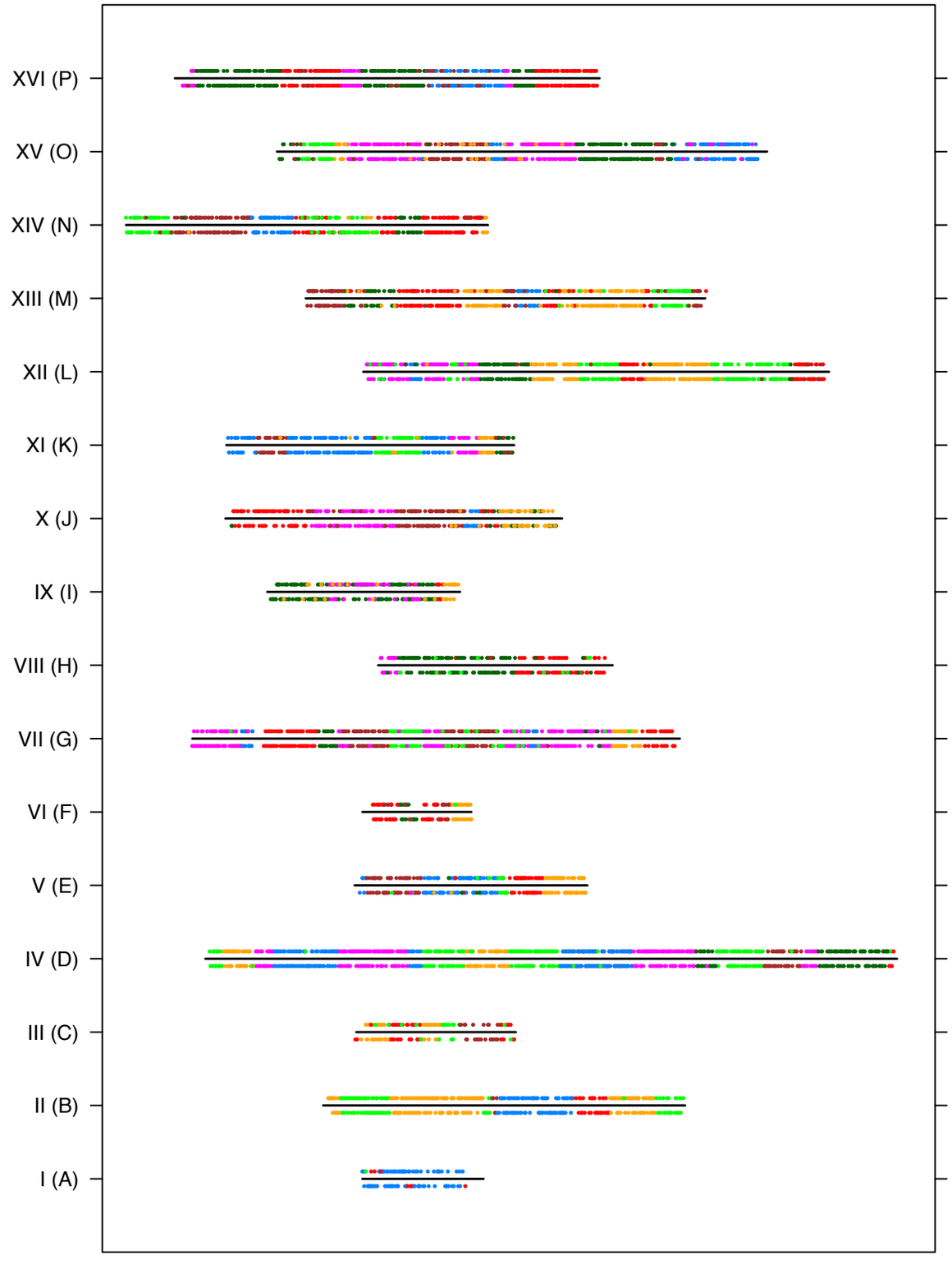
Z = the specific yeast gene that was deleted

e.g. YDL133W

Yeast genome has 16 chromosomes.

Each gene lives somewhere on one of these chromosomes.

Therefore, each deletion mutant is also associated with one yeast chromosome.



Y = a quantitative measure of growth

e.g. growth rate or # cells at study end

Z = the specific yeast gene that was deleted

e.g. YDL133W

X = the chromosome on which the gene is

e.g. "chromosome D"

So, 1 quantitative variable Y and 2 categorical variables X, Z. Kind of like Y = lifeExp and Z = country and X = continent in Gapminder data.

Minimum you need to know

- Measuring growth -- an example of a phenotype -- on thousands of different strains of yeast. This is the response.
- Each strain is characterized by lacking the DNA for one gene.
- Each gene is found on one of the sixteen chromosomes in the yeast genome. This is the (potential) explanatory variable.
- It is interesting, for biological and experimental reasons, to ask whether the distribution of 'growth after gene deletion' differs across the chromosomes.
- Gapminder analology: interesting to assess whether the distribution of life expectancy differs across the continents.

the typical "two groups" testing problem

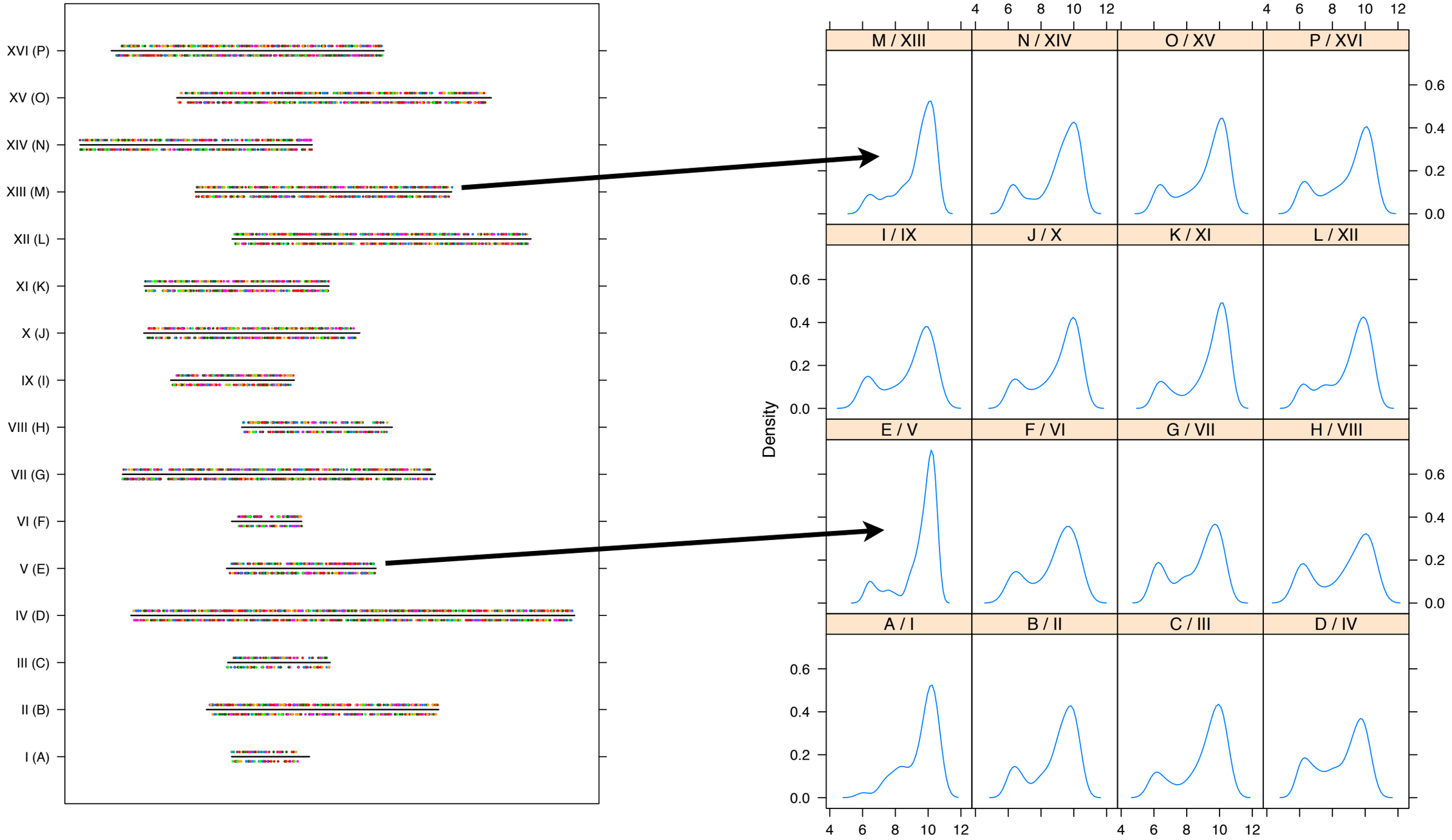
$$Y \mid X = x_i \sim F_i$$
$$Y \mid X = x_j \sim F_j$$

$$F_i = F_j \, ?$$

Y = measure of growth for mutant lacking a specific gene

X = chromosome on which this gene is found

Do you think there are systematic differences in the phenotype distribution across the 16 chromosomes?




```
densityplot(~pheno | chromoPretty, gDat,
            plot.points = FALSE, layout = c(4,4))
```

Data source: Giaever G, Flaherty P, Kumm J, Proctor M, Nislow C, et al. (2004) Chemogenomic profiling: identifying the functional interactions of small molecules in yeast. Proc Natl Acad Sci U S A 101: 793-798. Pubmed. DOI: 10.1073/pnas.0307490100

All of my code, data, figures, and results for this example: yeastDeletionGrowth

I have done some preprocessing and simplification of a subset of the data associated with the reference above. Specifically, we are looking at growth data on yeast deletion mutants growing in control conditions. Underlying dataset available supplementary information from the article above.

```
> str(gDat)                                  # 5666 observations
'data.frame':    5666 obs. of  4 variables:
 $ geneDel     : Factor w/ 5521 levels "YAL001C","YAL002W",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ chromo      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ chromoPretty: Factor w/ 16 levels "A / I","B / II",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ pheno       : num  9.39 9.4 10.38 10.54 8.65 ...

> peek(gDat)
     geneDel chromo chromoPretty     pheno
844  YDR008C      4       D / IV  9.686326
2691 YIL134W      9       I / IX  9.188266
3034 YJR091C     10        J / X 10.167940
3322 YKR024C     11       K / XI 10.530350
4270 YMR260C     13     M / XIII 10.338061
4359 YNL029C     14      N / XIV  9.855063
4973 YOR123C     15       O / XV  7.098627
```

Each row consists of

- geneDel = name of the gene that was deleted
- chromo = the associated chromosome (an integer between 1 and 16)
- chromoPretty = a prettier version of the chromosome (more suitable for labeling in tables and figures)
- pheno = a growth phenotype (due to experimental realities and pre-processing, the units are meaningless, i.e. don't expect to see a cell count here)

How many observations? How many genes? How many observations per gene?

```
> str(gDat)                          # 5666 observations
'data.frame': 5666 obs. of  4 variables:
 $ geneDel     : Factor w/ 5521 levels "YAL001C","YAL002W",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ chromo      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ chromoPretty: Factor w/ 16 levels "A / I","B / II",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ pheno       : num   9.39  9.40 10.38 10.54  8.65 ...

> table(table(gDat$geneDel))

   1    2
5376  145
```

This is puzzling.

5666 observations
5521 unique genes
5376 genes contribute 1 observation, 145 contribute 2
5376 + 145 = 5521 ✓
5376 + 2*145 = 5666 ✓

table() is a useful function
sometimes makes sense: table(table(...))

"Exhaustion and checking"

- The principle of exhaustion reassures the reader (and analyst!) that all cases are accounted for.
- GOOD: "Of the 84 patients with myocardial infarction, 22 had subendocardial infarction and 62 transmural infarction, with 78 discharged alive -- a mortality in the hospital of 7 percent."
- BAD: "... 78 patients were discharged alive, leading to an in-hospital mortality rate of 7 percent.

GOOD:
22 myo + 62 subendo = 84 ✓
84 - 78 = 6 died
6/84 = 0.07 ✓

BAD:
6/n = 0.07
(reader jots in the margin ...)
6 = 0.07n
n = 6/0.07
n = 84

"Exhaustion and checking"

- Much easier for a reader to accept / confirm results that are complete and transparent.
  - Don't make the reader do irritating arithmetic.
  - Don't (seem to) contradict yourself!
  - Don't seem like you're hiding something!
- Build your credibility with your audience, i.e. that you are careful, thorough, and honest.
- Presents a golden opportunity for you to catch errors and confusion, before you start the analysis.
- Reference: "Writing About Numbers" by Frederick Mosteller in Medical Uses of Statistics, 2nd edition, Bailer and Mosteller (eds), NEJM Books, 1992.

How many observations per chromosome?
How many unique genes?

Chromosome	# observations	# unique genes	# replicated genes
A / I	87	87	0
B / II	370	370	0
C / III	135	132	3
D / IV	835	754	81
E / V	248	248	0
F / VI	105	105	0
G / VII	561	512	49
H / VIII	221	221	0
I / IX	203	202	1
J / X	326	325	1
K / XI	302	302	0
L / XII	500	499	1
M / XIII	442	437	5
N / XIV	373	372	1
O / XV	506	503	3
P / XVI	452	452	0

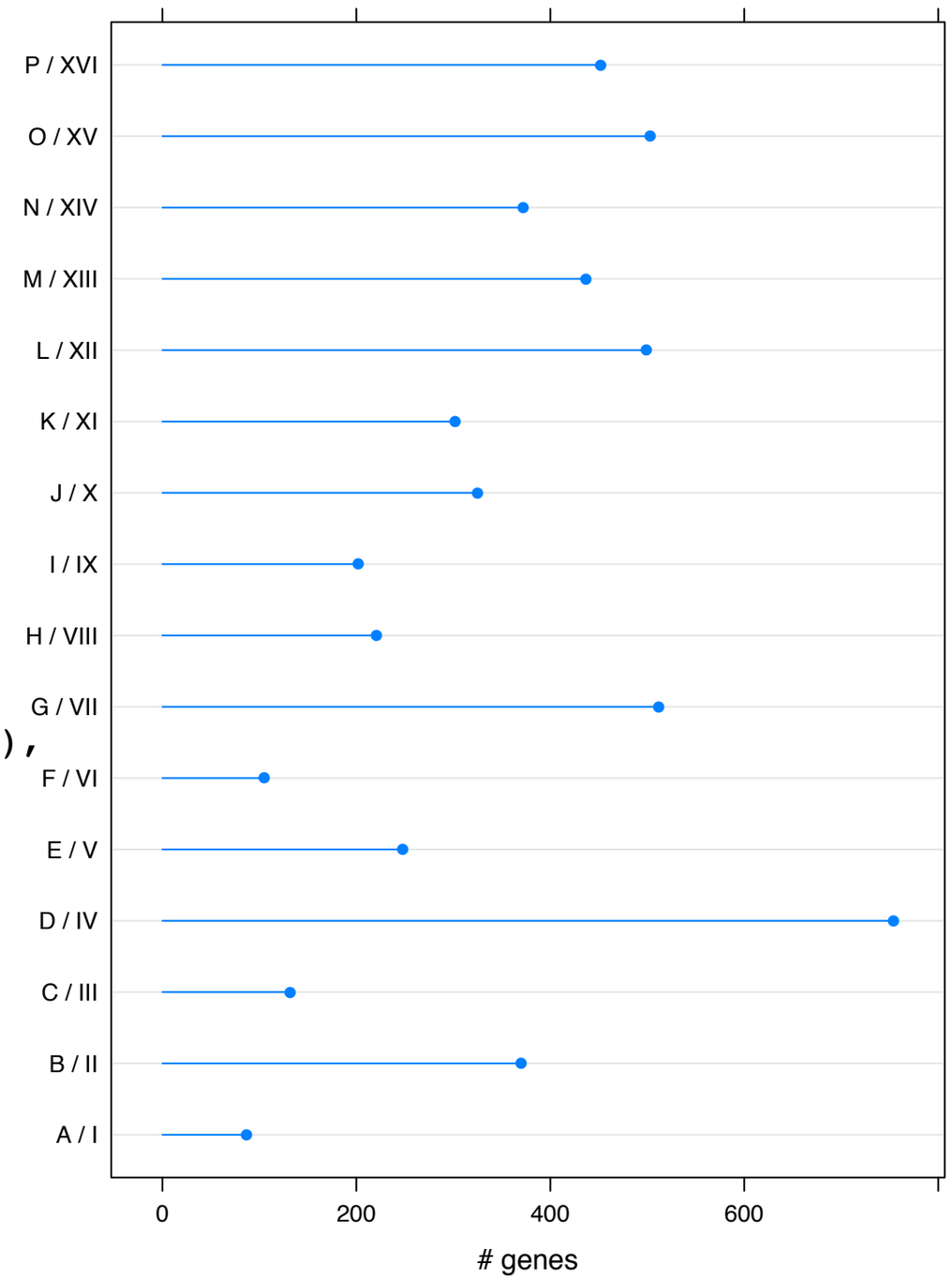
```
gFreqTable <-
  data.frame(obsCount = table(gDat$chromoPretty))

## what if we want to count the number of unique genes
## on each chromosome?
## 'tapply' applies a function to a ragged array

gFreqTable$uniqGeneCount <-
  cbind(tapply(gDat$geneDel, gDat$chromoPretty,
               function(x) {return(length(unique(x)))}))

## facilitate putting this table in a report
write.table(gFreqTable,
            file = paste(whereAmI, "chromoFreq.txt", sep = ""),
            quote = FALSE, row.names = FALSE, sep = "\t")
```

(easily copied and pasted from “chromoFreq.txt”)

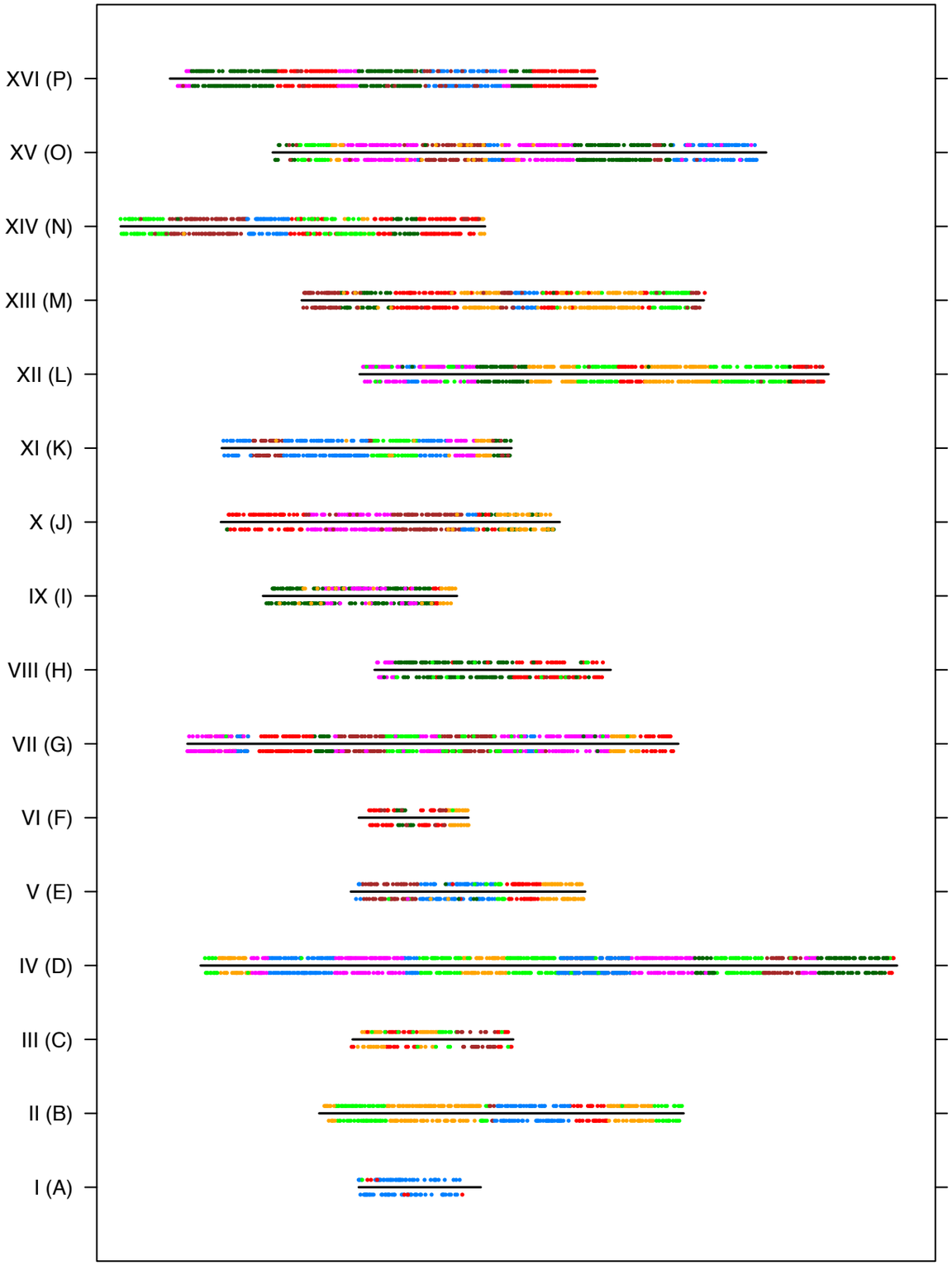
Getting to the bottom of the duplicated gene deletions

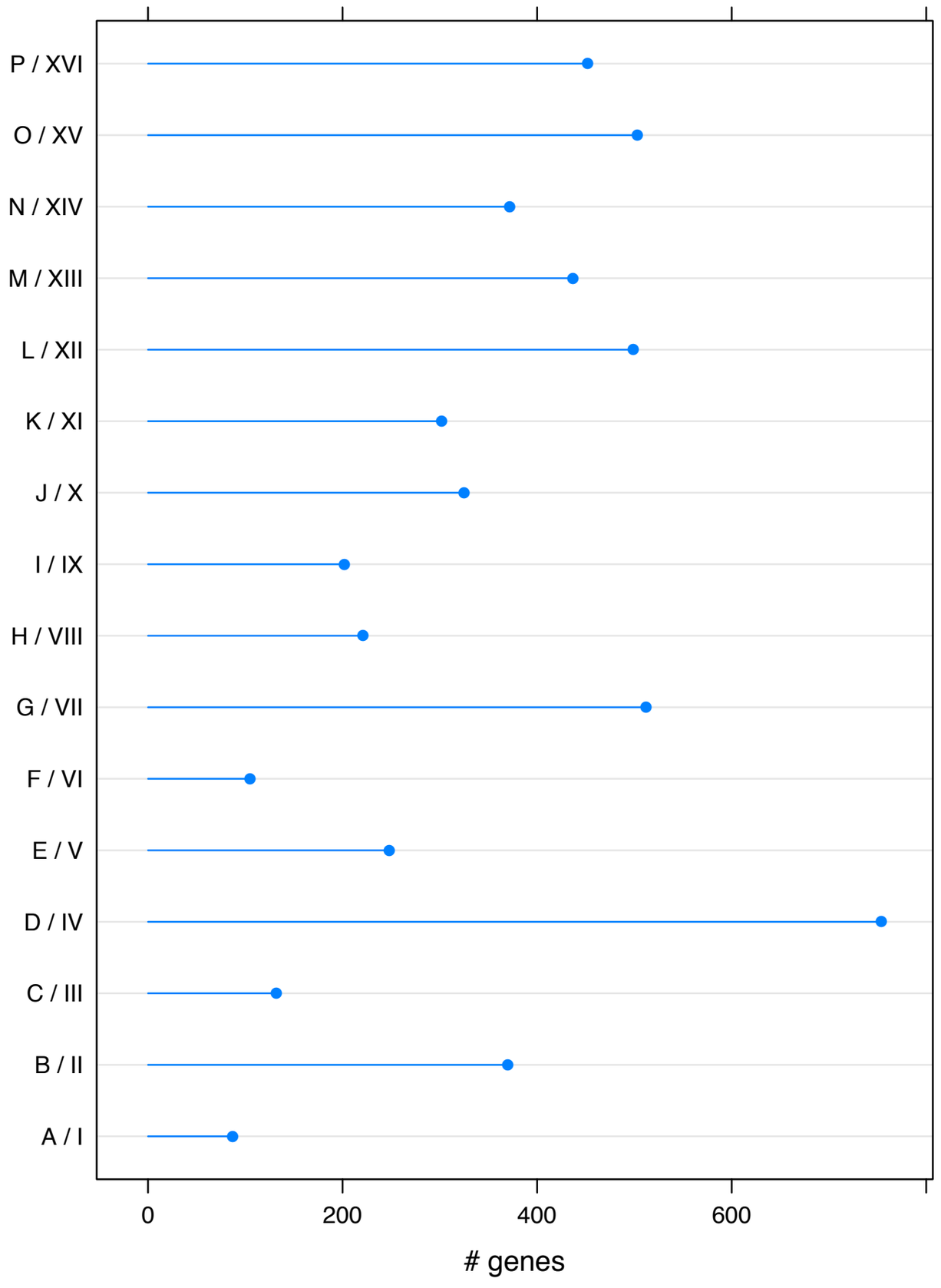
- See code for details
- Short version:
  - When there are 2 rows for a gene deletion, such as YCL038C, it turns out the observed phenotypes are exactly the same
  - So, it's a not a legitimate, independent observation of the growth phenotype
  - Immediate implication: I've eliminated these duplicates.
- hDat nows holds the clean, duplicate-free data

```
> str(hDat)
'data.frame':  5521 obs. of  4 variables:
 $ geneDel     : Factor w/ 5521 levels "YAL001C","YAL002W",..: 1 2 3 4 5 6 7 8..
 $ chromo      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ chromoPretty: Factor w/ 16 levels "A / I","B / II",..: 1 1 1 1 1 1 1 1 1 1 ..
 $ pheno       : num  9.39 9.4 10.38 10.54 8.65 ...

> peek(hDat)
     geneDel chromo chromoPretty    pheno
190  YBL102W      2       B / II 9.285750
917  YDR089W      4       D / IV 9.528659
1040 YDR185C      4       D / IV 7.079669
1969 YGL201C      7      G / VII 9.754082
2118 YGR046W      7      G / VII 9.262812
3175 YKL085W     11       K / XI 9.479903
3622 YLR176C     12      L / XII 7.359638

> write.table(hDat,
+             jPaste(whereAmI, "data/hDat.txt"),
+             quote = FALSE, row.names = FALSE, sep = "\t")

> dotplot(table(hDat$chromoPretty),
+         origin = 0, type = c("p", "h"),
+         xlab = "# genes")

> dev.print(pdf,
+           jPaste(whereAmI, "figs/genesPerChromoDotplot.pdf"),
+           width = 6, height = 8)
quartz
     2
```

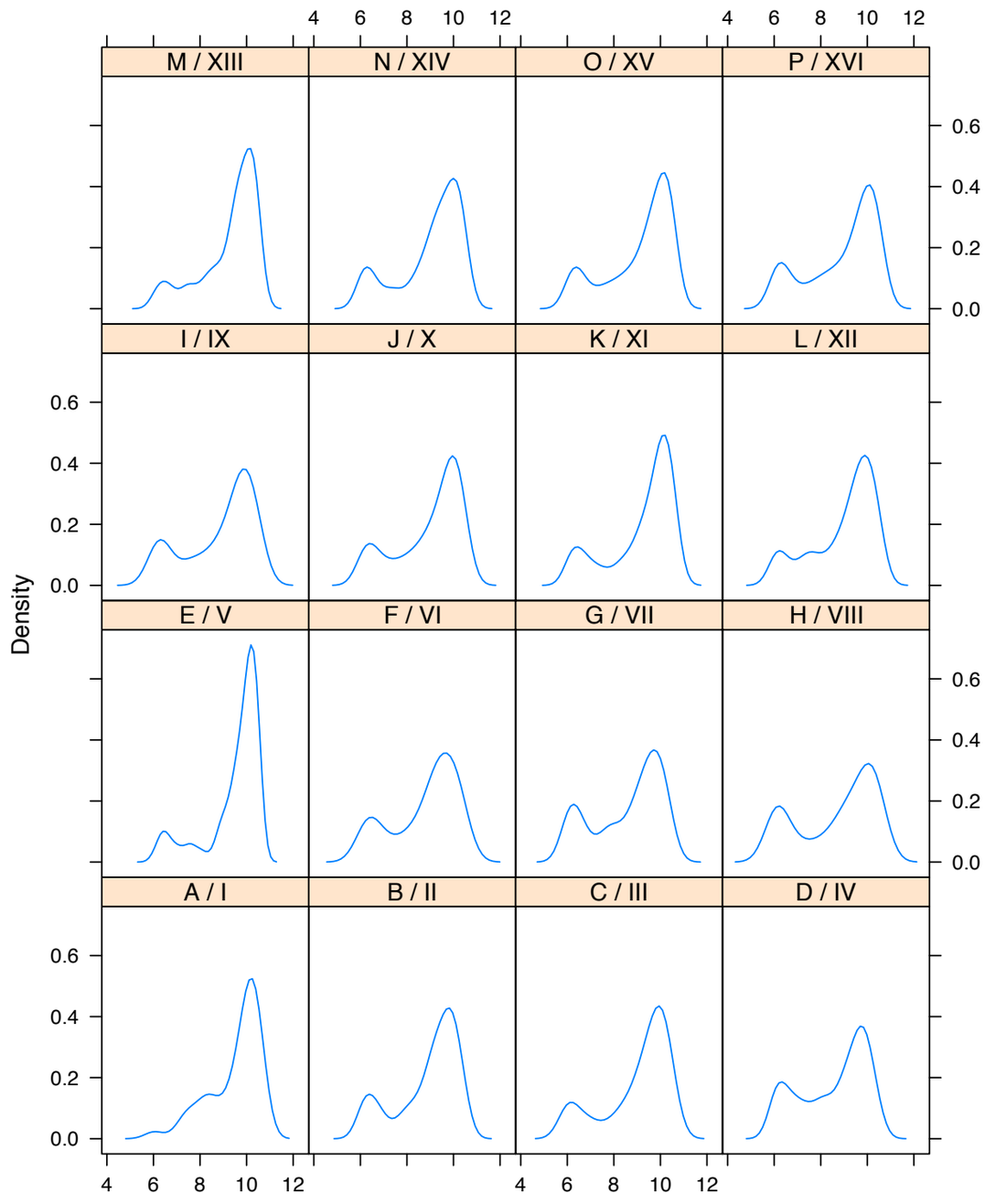


sanity check of # genes on each chromosome





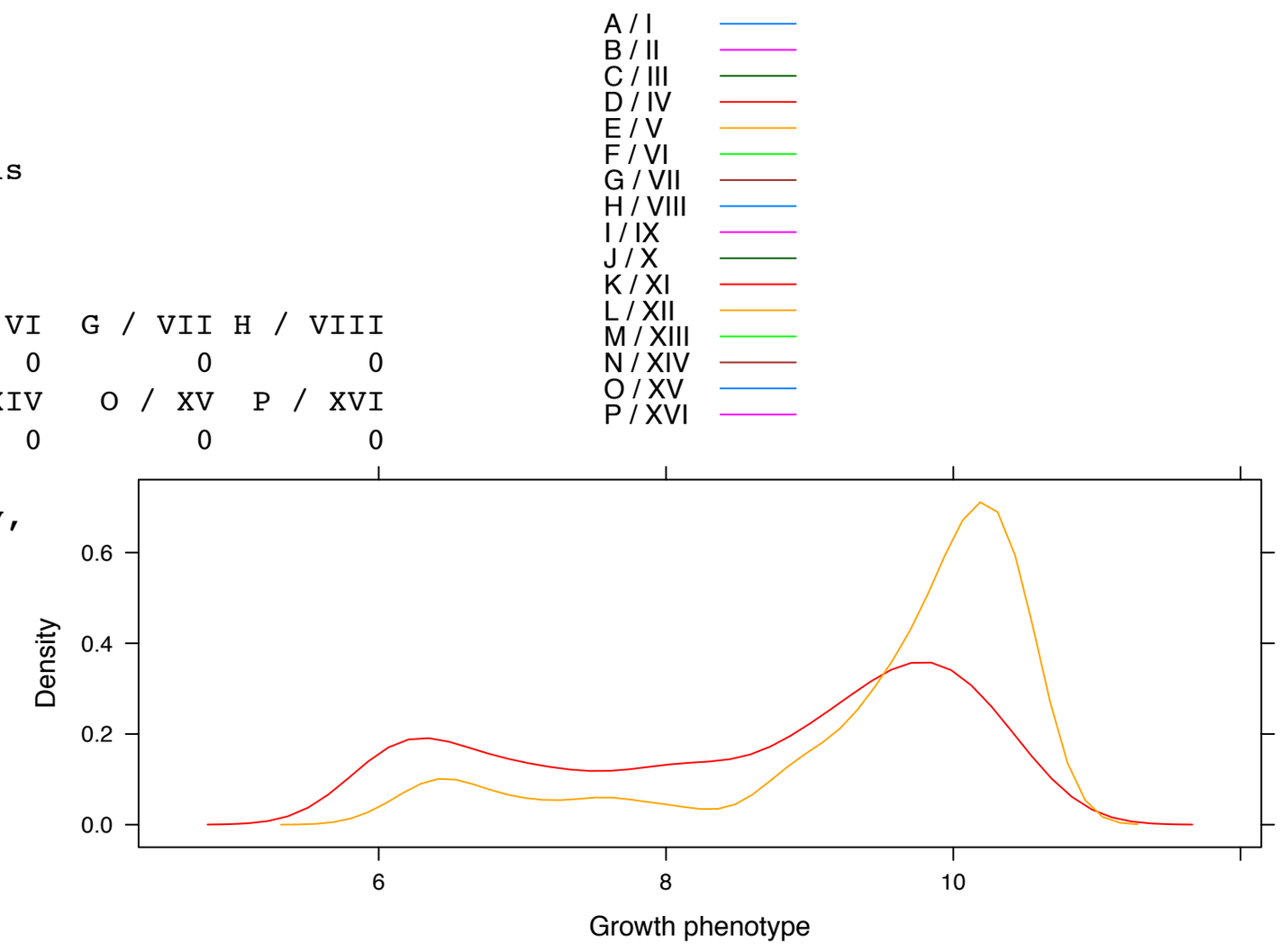
```
> densityplot( ~ pheno | chromoPretty, hDat,
+              plot.points = FALSE, layout = c(4,4),
+              xlab = "Growth phenotype")
> dev.print(pdf,
+           jPaste(whereAmI, "figs/phenoDensityplotByChromo4by4.pdf"),
+           width = 8, height = 6)
quartz
     2
```



Real differences in distribution?

I'll focus on comparing two chromosomes: 4 vs. 5.

```
> iDat <- subset(hDat, chromo %in% 4:5)

> str(iDat)                     # 'data.frame': 1002 obs. of  4 variables
'data.frame':    1002 obs. of  4 variables:
 $ geneDel     : Factor w/ 5521 levels "YAL001C","YAL002W",..: 590 591 592 593..
 $ chromo      : int  4 4 4 4 4 4 4 4 4 4 ...
 $ chromoPretty: Factor w/ 16 levels "A / I","B / II",..: 4 4 4 4 4 4 4 4 4 4 ..
 $ pheno       : num  9.78 8.23 9.82 7.62 8.24 ...

> table(iDat$chromo)

  4   5
754 248

> ## highlighting issues around unused factor levels

> table(iDat$chromoPretty)

   A / I    B / II   C / III    D / IV     E / V    F / VI   G / VII H / VIII
       0         0         0       754       248         0         0         0
  I / IX     J / X    K / XI   L / XII M / XIII   N / XIV    O / XV   P / XVI
       0         0         0         0         0         0         0         0

> densityplot( ~ pheno, iDat, groups = chromoPretty,
+              plot.points = FALSE, auto.key = TRUE,
+              xlab = "Growth phenotype")
```

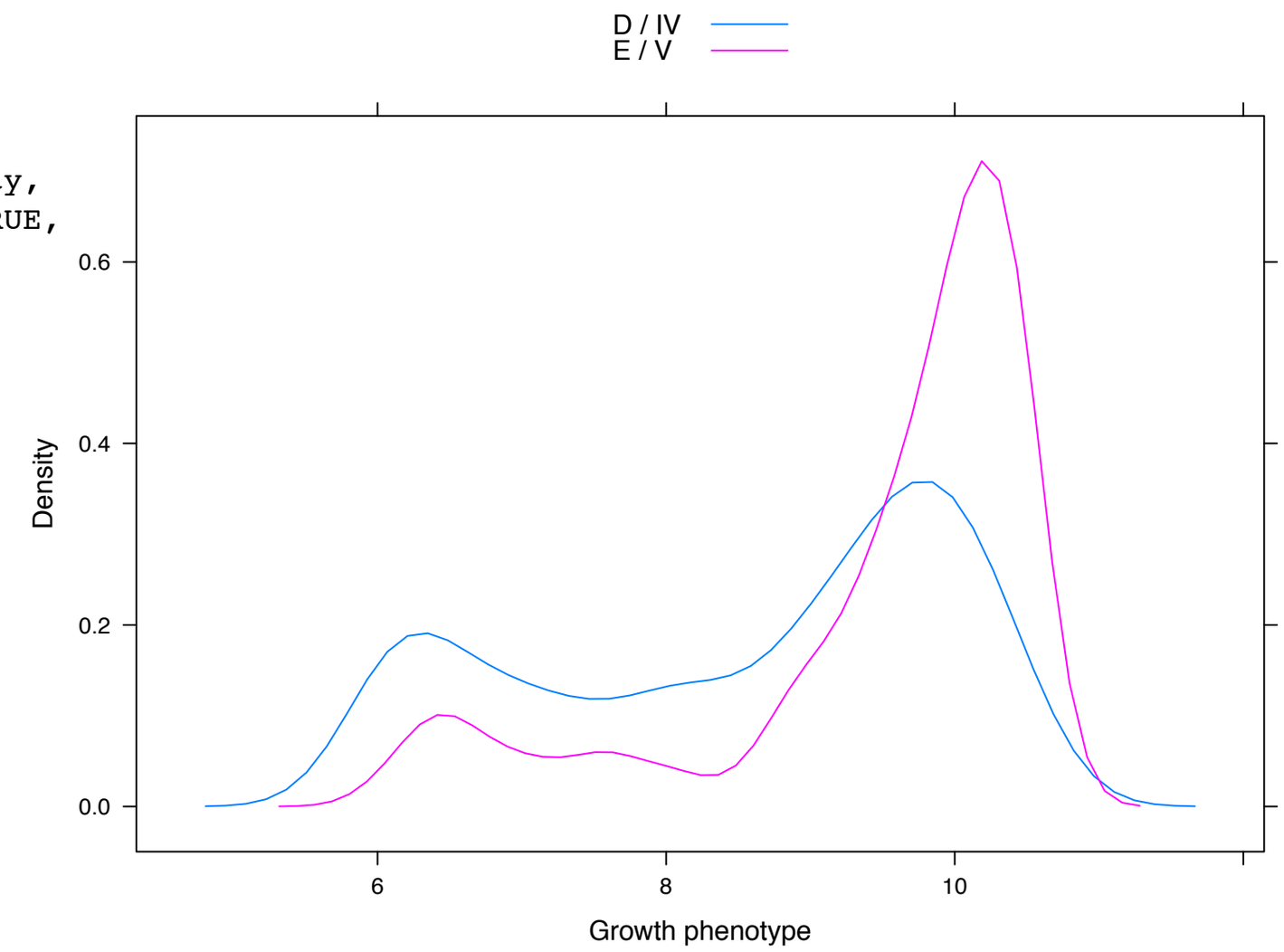


Real differences in distribution?

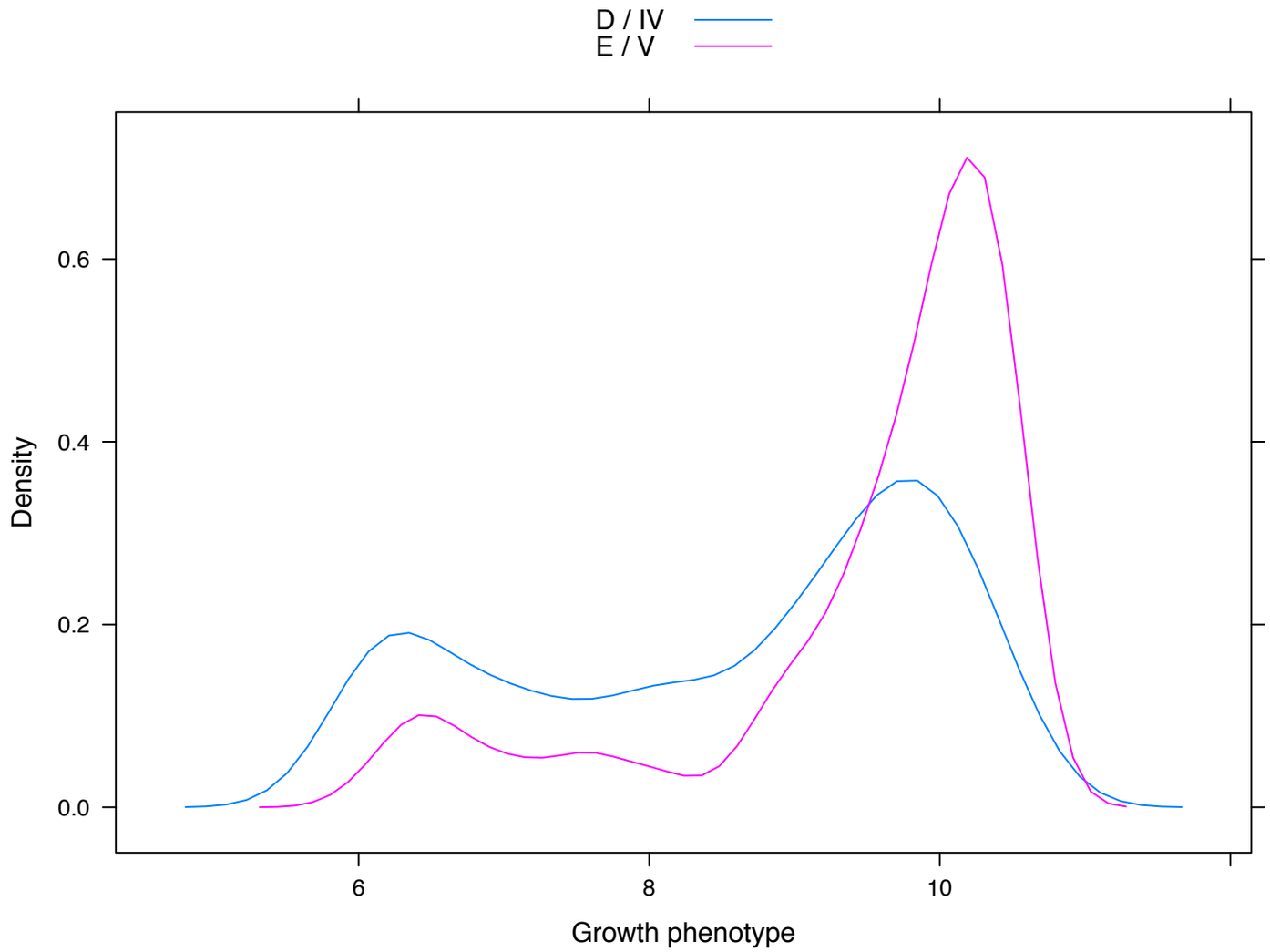
I'll focus on comparing two chromosomes: 4 vs. 5.

```
> ## dropping unused factor levels

> iDat <- droplevels(iDat)

> levels(iDat$chromoPretty)
[1] "D / IV" "E / V"

> table(iDat$chromoPretty)

D / IV  E / V
   754    248
```

```
> ## remake superposed plot
> densityplot(~pheno, iDat, groups = chromoPretty,
+             plot.points = FALSE, auto.key = TRUE,
+             xlab = "Growth phenotype")
```



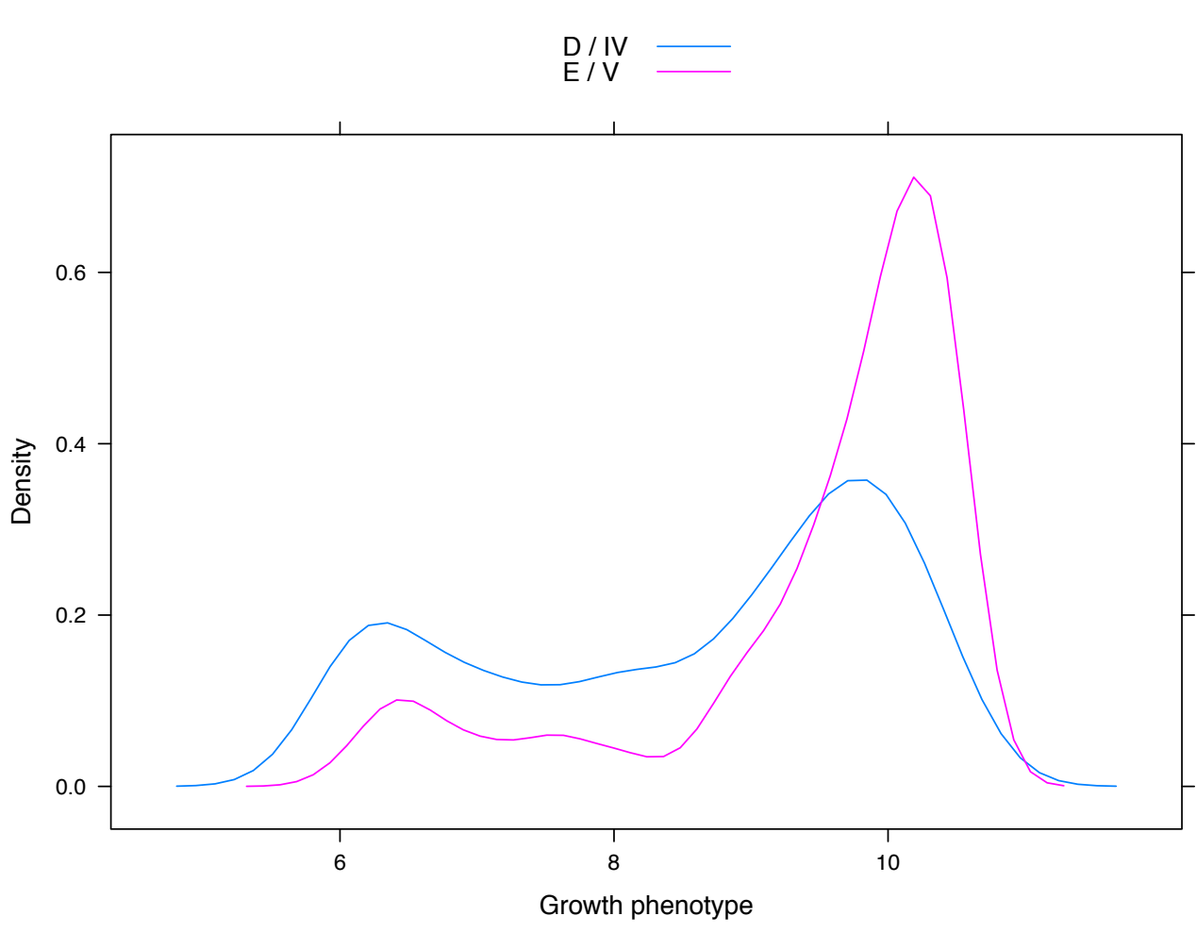
Real differences in distribution?

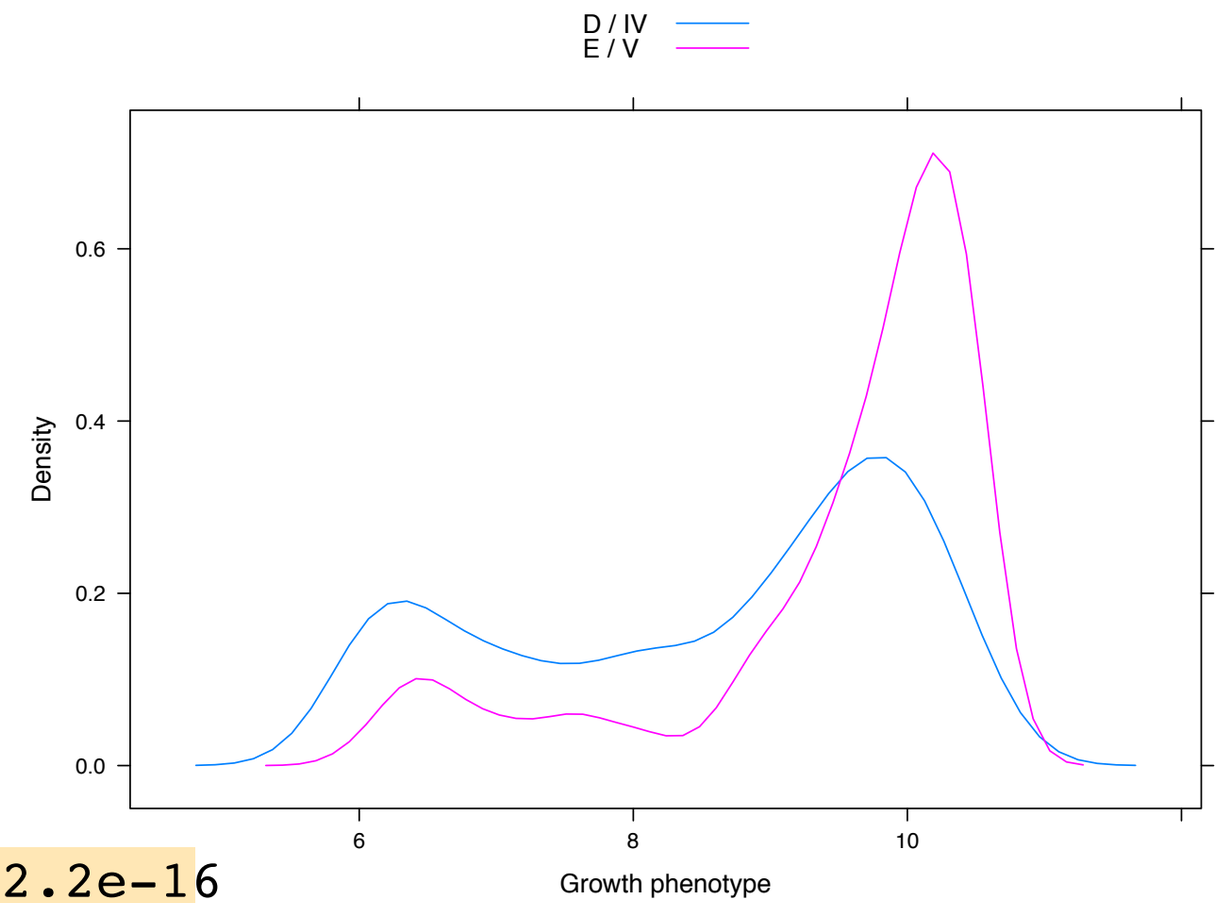


How would you assess?

Tests that address our question

- t test
- Wilcoxon test, aka Mann-Whitney here
- Kolmogorov-Smirnov test, 2 sample version
- Chi-square test of homogeneity
- I'm sure there are others ....






```
> t.test(pheno ~ chromo, iDat)

    Welch Two Sample t-test

data:  pheno by chromo
t = -10.0549, df = 512.318, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.1101599 -0.7472465
sample estimates:
mean in group 4 mean in group 5
       8.548879        9.477582

> wilcox.test(pheno ~ chromo, iDat)

    Wilcoxon rank sum test with continuity correction

data:  pheno by chromo
W = 53853, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

t test and Wilcoxon test

- t.test ... I assume all are familiar
  - default in R is to NOT assume common variance, i.e. to perform Welch's t test
  - suitability?
- Wilcoxon test is based on ranks, therefore is nonparametric
  - Rank all the data, ignoring the grouping variable
  - Compute the sum of the ranks for one group
  - Null distribution of this can be worked out / approximated
  - Also called Mann-Whitney test

Kolmogorov-Smirnov test (two sample)

- Hypothesis: $F_i = F_j$, i.e. distributions are same
- Estimate each CDF with the empirical CDF (ECDF)

$$\hat{F}_i(x) = \frac{1}{n}\sum_k I[x_{i,k} \le x]$$

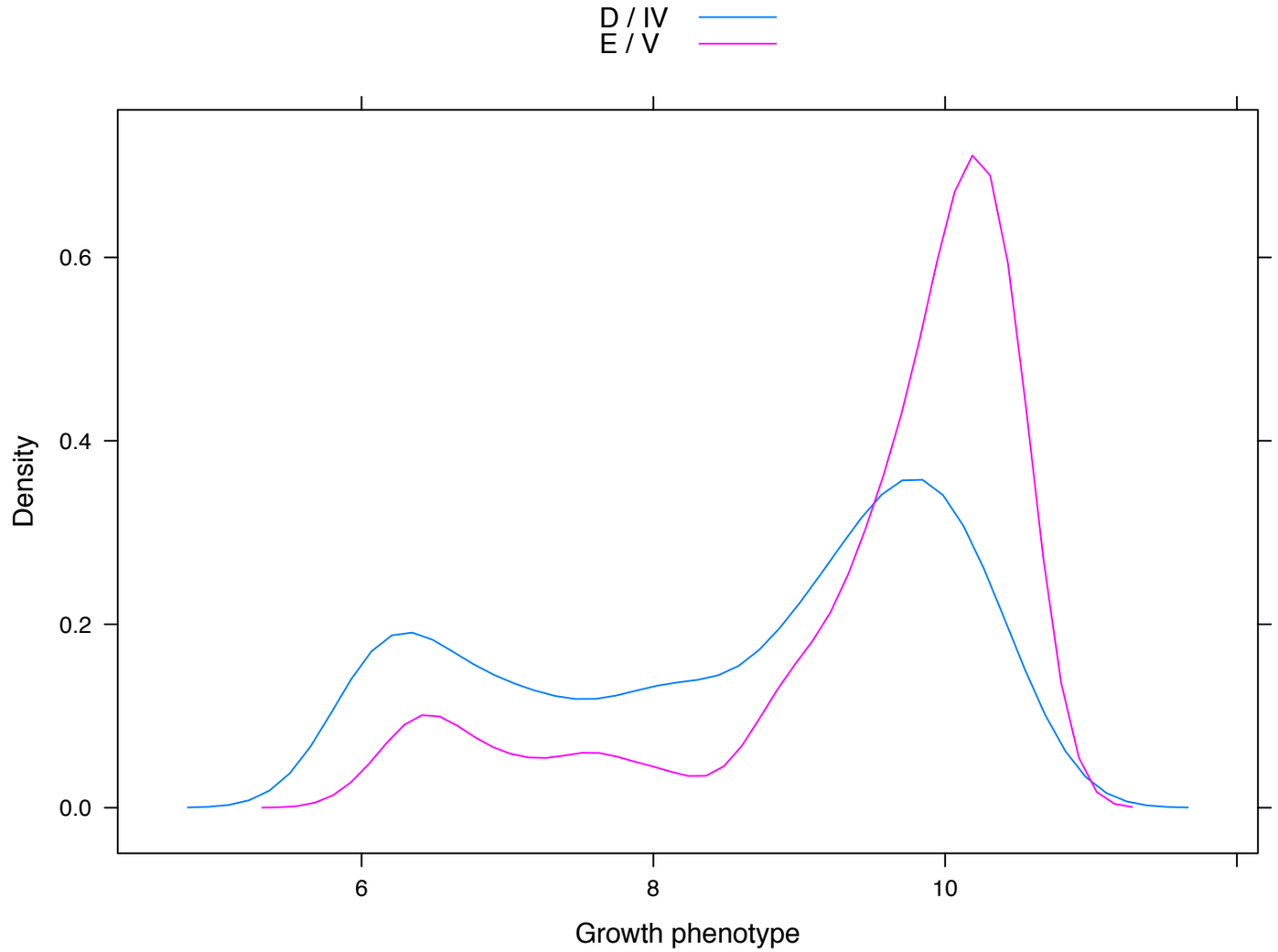
- Test statistic is the maximum of the absolute difference between the ECDFs

$$\max\left|\hat{F}_i(x) - \hat{F}_j(x)\right|$$

- Null distribution does not depend on F_i, F_j (!)
- (I'm suppressing detail here.)

Kolmogorov-Smirnov test (two sample)

```
> ## sadly, Kolmogorov-Smirnov test has no formula interface
> jDat <- split(iDat$pheno, iDat$chromoPretty)
> str(jDat)
List of 2
 $ D / IV: num [1:754] 9.78 8.23 9.82 7.62 8.24 ...
 $ E / V : num [1:248] 10.57 8.62 10.6 10.57 8.93 ...

> ks.test(x = jDat[[1]], y = jDat[[2]])

    Two-sample Kolmogorov-Smirnov test

data:  jDat[[1]] and jDat[[2]]
D = 0.3349, p-value < 2.2e-16
alternative hypothesis: two-sided
```



Chi-square test of homogeneity

- If $F_i = F_j$, then ... if we divide the support into bins, the observed relative frequency of these bins should be approximately the same in the two samples

- Obviously, choice of the bin boundaries is a big issue. Certainly true in terms of the theory. In practice, this should work fairly well most of the time.

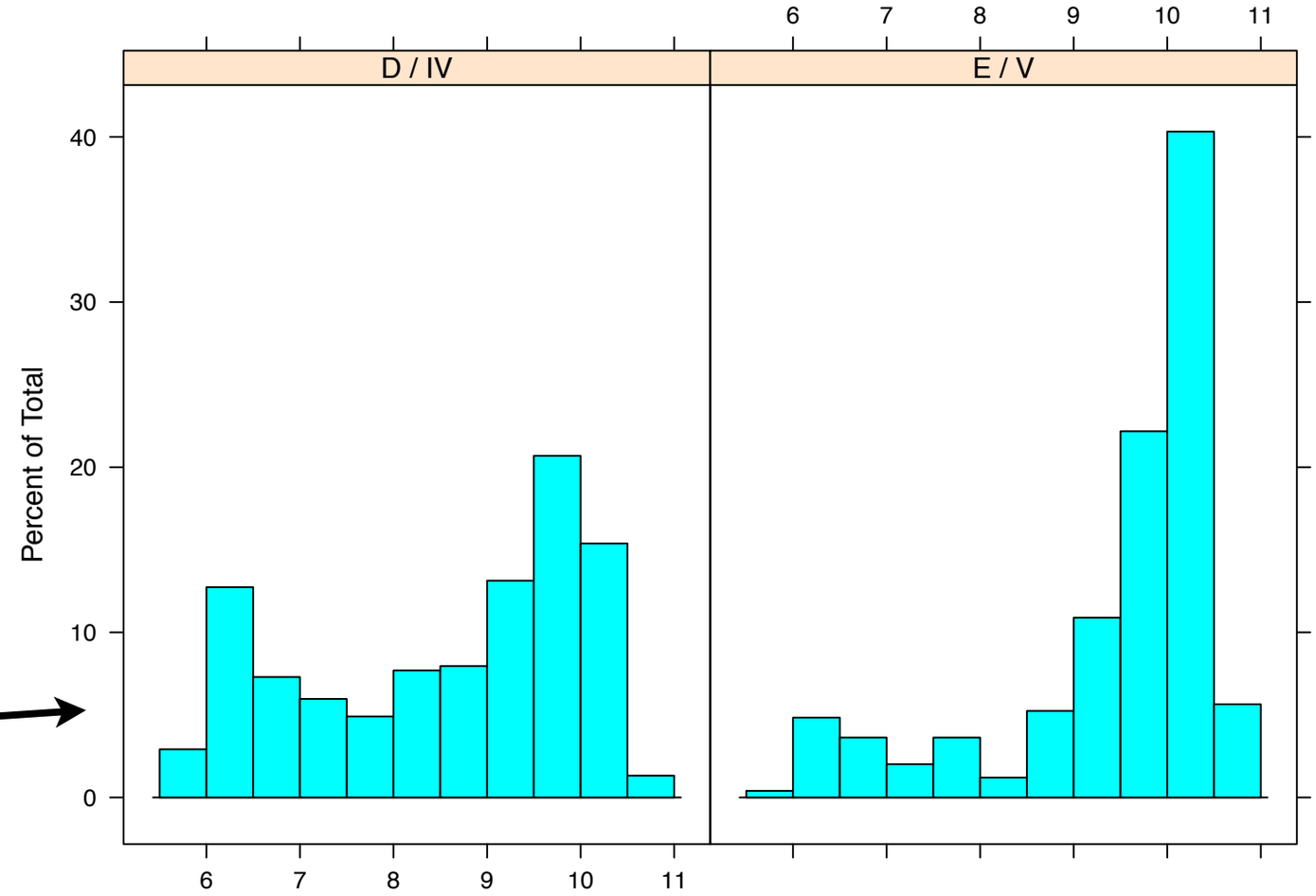
- A 'pooled' estimate of the expected relative frequencies is obtained by ignoring the grouping variable.

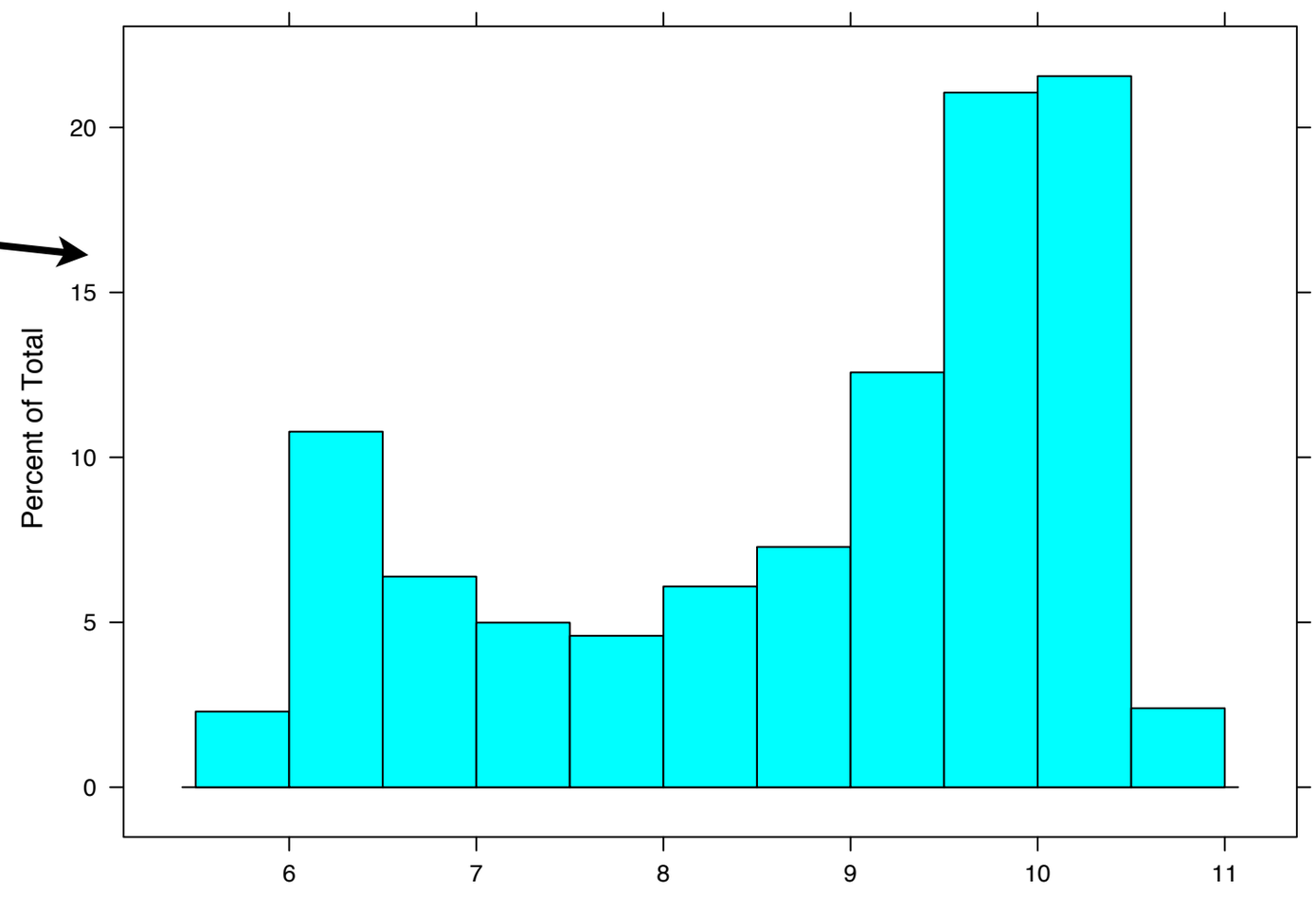
- Conventional chi square test statistic arises from summing the usual terms:

  - (observed - expected)^2 / expected

Could the observed bin counts (top) be random draws from one common underlying distribution (bottom)?

```
> addmargins(jCounts)
           chr4 chr5  Sum
5.5 - 6      22    1   23
6 - 6.5      96   12  108
6.5 - 7      55    9   64
7 - 7.5      45    5   50
7.5 - 8      37    9   46
8 - 8.5      58    3   61
8.5 - 9      60   13   73
9 - 9.5      99   27  126
9.5 - 10    156   55  211
10 - 10.5   116  100  216
10.5 - 11    10   14   24
Sum         754  248 1002
```





```
> (jBreaks <- hist(iDat$pheno)$breaks)    # plot is nice by-product!
 [1]  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0

> jCounts <- sapply(jDat, function(jPheno) {
+   hist(jPheno, breaks = jBreaks, plot = FALSE)$counts
+ })

> rownames(jCounts) <-
+   paste(jBreaks[-length(jBreaks)], jBreaks[-1], sep = " - ")

> addmargins(jCounts)
           chr4 chr5  Sum
5.5 - 6      22    1   23
6 - 6.5      96   12  108
6.5 - 7      55    9   64
7 - 7.5      45    5   50
7.5 - 8      37    9   46
8 - 8.5      58    3   61
8.5 - 9      60   13   73
9 - 9.5      99   27  126
9.5 - 10    156   55  211
10 - 10.5   116  100  216
10.5 - 11    10   14   24
Sum         754  248 1002
```

cut() is another useful function for making a quantitative variable into a factor

addmargins(), prop.table(), etc. helpful for working with contingency tables

```
> chisq.test(jCounts)

    Pearson's Chi-squared test

data:  jCounts
X-squared = 110.4444, df = 10, p-value < 2.2e-16
```

OK, we just conducted 4 different hypothesis tests. It would be nice to gather them together for comparison, no?

Especially when we scale up to other pairs of chromosomes, where the results might not be so unanimous.

Get to know the result objects

```
> tTestResult <- t.test(pheno ~ chromo, iDat)

> class(tTestResult)                          # "htest"
[1] "htest"

> names(tTestResult)
[1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
[6] "null.value"  "alternative" "method"      "data.name"

> tTestResult$statistic
        t
-10.05494

> tTestResult$p.value
[1] 7.906832e-22

> tTestResult$estimate
mean in group 4 mean in group 5
       8.548879        9.477582

> tTestResult$conf.int
[1] -1.1101599 -0.7472465
attr(,"conf.level")
[1] 0.95
```

In addition to str(), class() and names() will help you understand the basics about most objects.

Once you understand the structure, you're ready to extract the good stuff.

Get to know the result objects

```
> ksTestResult <- ks.test(x = jDat[[1]], y = jDat[[2]])

> class(ksTestResult)
[1] "htest"

> names(ksTestResult)                          # quite different from t test
[1] "statistic"   "p.value"     "alternative" "method"      "data.name"

> ksTestResult$statistic
        D
0.3348592

> ksTestResult$p.value
[1] 0

> ksTestResult$estimate                        # not present
NULL

> ksTestResult$conf.int                        # not present
NULL
```

Contrast Kolmogorov-Smirnov output with the t test.

make a list and extract from it

```
> jTestResults <-
+   list(t = t.test(pheno ~ chromo, iDat),
+        wilcox = wilcox.test(pheno ~ chromo, iDat),
+        ks = ks.test(x = jDat[[1]], y = jDat[[2]]),
+        chisq = jChiSquareTest(jDat[[1]], jDat[[2]]))

> sapply(jTestResults, class)              # all htest
       t   wilcox       ks    chisq
"htest" "htest" "htest" "htest"

> (jPvals <- sapply(jTestResults, function(yo) return(yo$p.value)))
           t       wilcox           ks        chisq
7.906832e-22 1.154823e-23 0.000000e+00 4.340871e-19
```

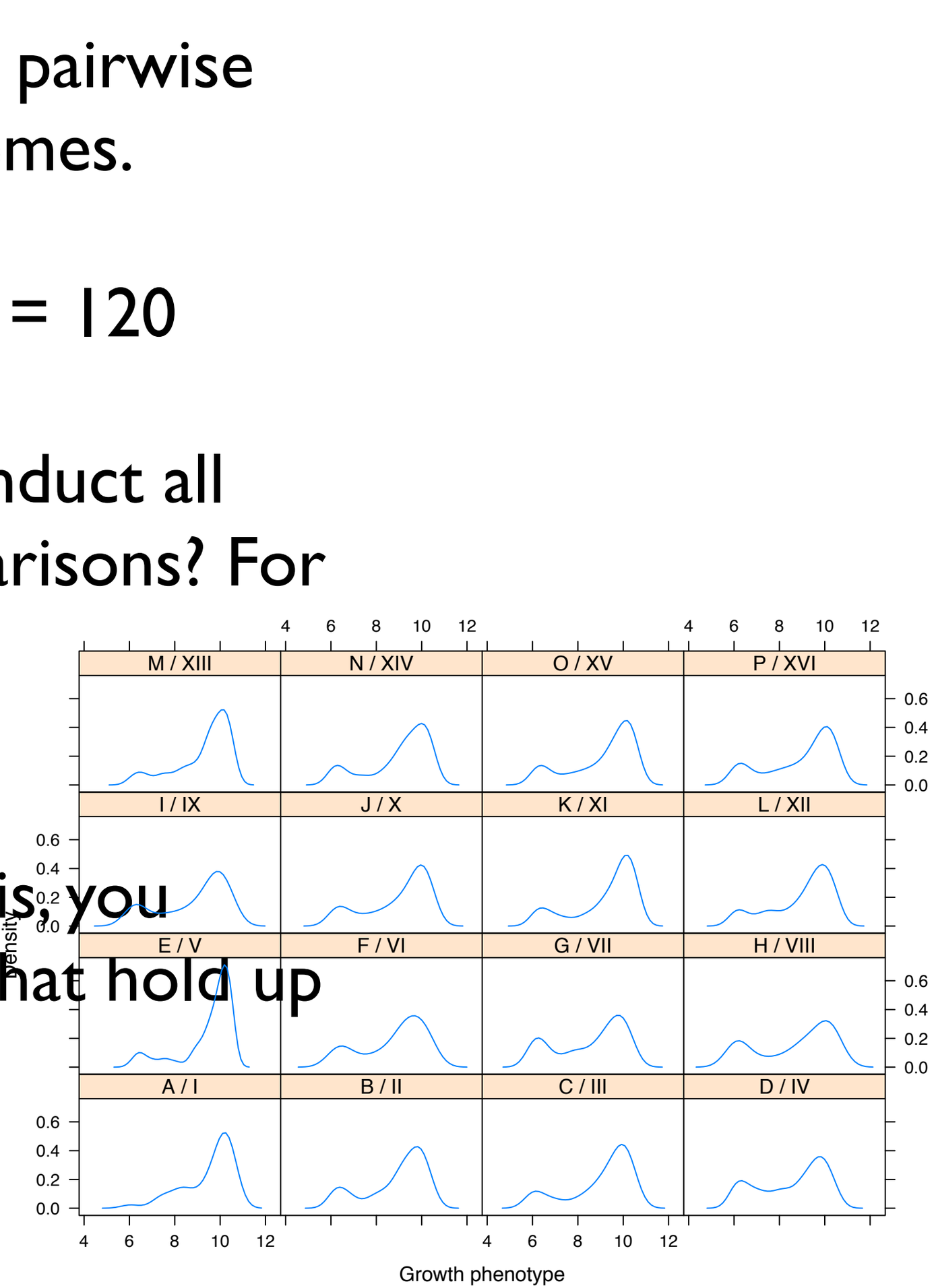
- *lists* can hold just about anything. Here, the objects are hypothesis test results. All have class "htest".
- You don't even really need to understand what that means, as long as you realize you can use sapply() to pull out the p-values of each component of the list.
- Sometimes key info from non-rectangular objects can be extracted and stored more easily than the entire object.

Now, consider all possible pairwise comparisons of chromosomes.

16 choose 2 = 16 * 15 / 2 = 120

Wouldn't it be cool to conduct all possible two group comparisons? For any one of the tests we're considering?

In exploratory data analysis, you want to find conclusions that hold up under various choices of methodology!



All possible pairs -- disclaimer

- This isn't so elegant statistically -- and unfortunately I see similar things done frequently.
- Remember, we have better ways to systematically answer the question "Is the distribution of phenotype common to all chromosomes?"
  - ANOVA, for starters (although not a great idea here)
  - bootstrap approaches
  - random or mixed effects models
- BUT, maybe this is sometimes a good idea. Here, it still presents a good learning opportunity.

I enumerate all possible pairs of chromosomes and apply all 4 “two groups” tests and gather all of the p-values.

```
> str(tgtRes)
'data.frame':   120 obs. of  6 variables:
 $ chromoA: int  1 1 1 1 1 1 1 1 1 1 ...
 $ chromoB: int  2 3 4 5 6 7 8 9 10 11 ...
 $ t      : num  1.04e-05 4.86e-03 2.67e-11 8.66e-01 6.26e-05 ...
 $ wilcox : num  7.29e-07 2.10e-03 5.39e-11 7.77e-01 1.17e-05 ...
 $ ks     : num  2.53e-06 1.87e-02 1.26e-08 7.68e-01 2.98e-05 ...
 $ chisq  : num  2.19e-07 5.55e-02 3.27e-09 6.26e-03 3.54e-04 ...
> peek(tgtRes)
    chromoA chromoB            t       wilcox           ks         chisq
23        2      10 5.077492e-01 1.260279e-01 4.265804e-02 3.055263e-02
28        2      15 2.749797e-02 1.080776e-04 3.271894e-06 1.146343e-06
43        4       5 7.906832e-22 1.154823e-23 0.000000e+00 4.340871e-19
65        5      16 7.659286e-08 7.746156e-07 7.007464e-06 2.444194e-04
74        6      15 3.601006e-02 2.847043e-03 6.857203e-03 3.464897e-02
90        8      14 1.344403e-02 1.278636e-01 4.284638e-02 5.906893e-03
119      14      16 4.117154e-01 9.092607e-01 2.626741e-01 9.915321e-02
```

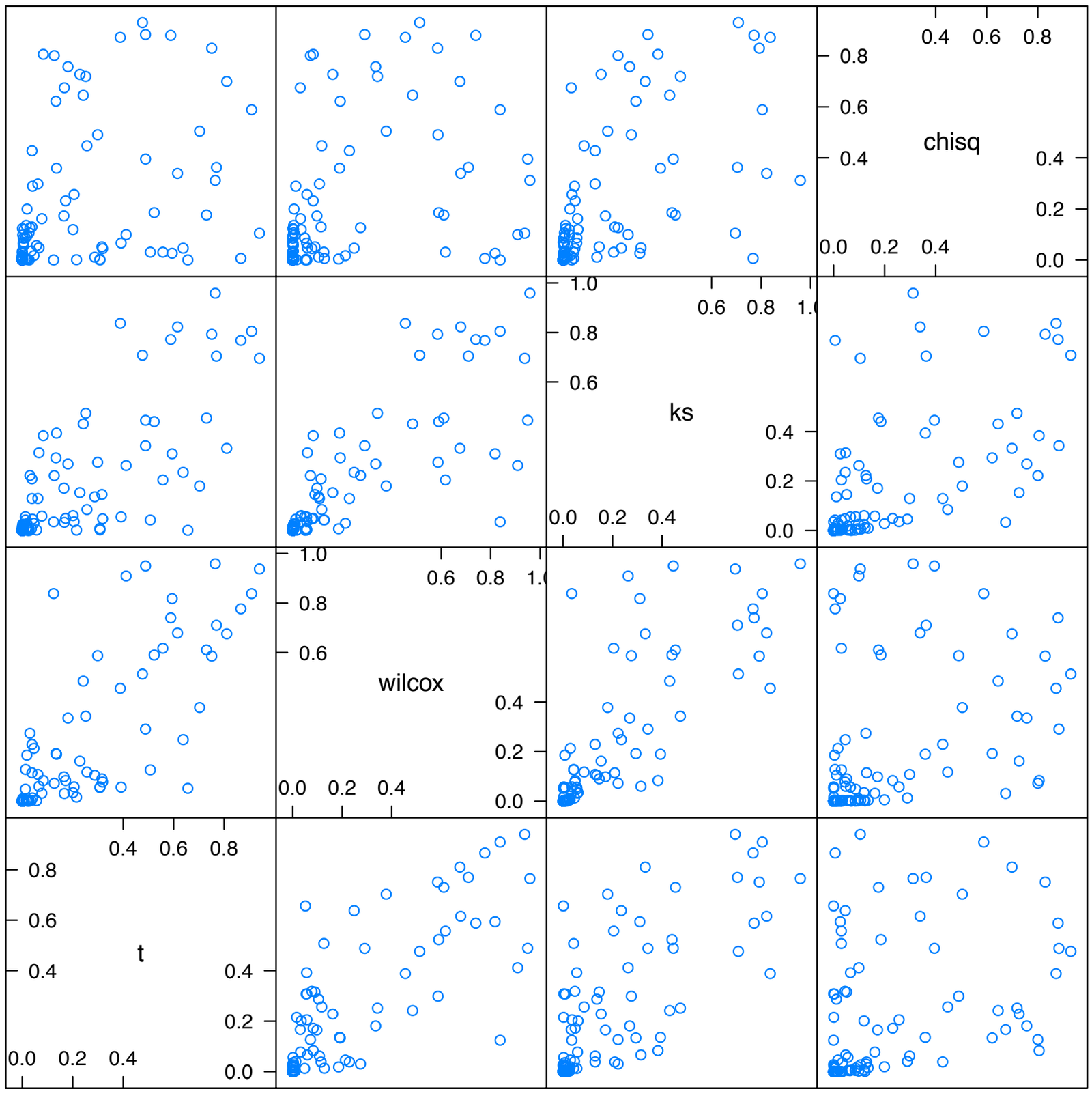
How to do this with apply() instead of an explicit loop -- or, even worse, a double loop? See code, if you are interested!

Let's make a scatterplot matrix of the p-values from the various tests.

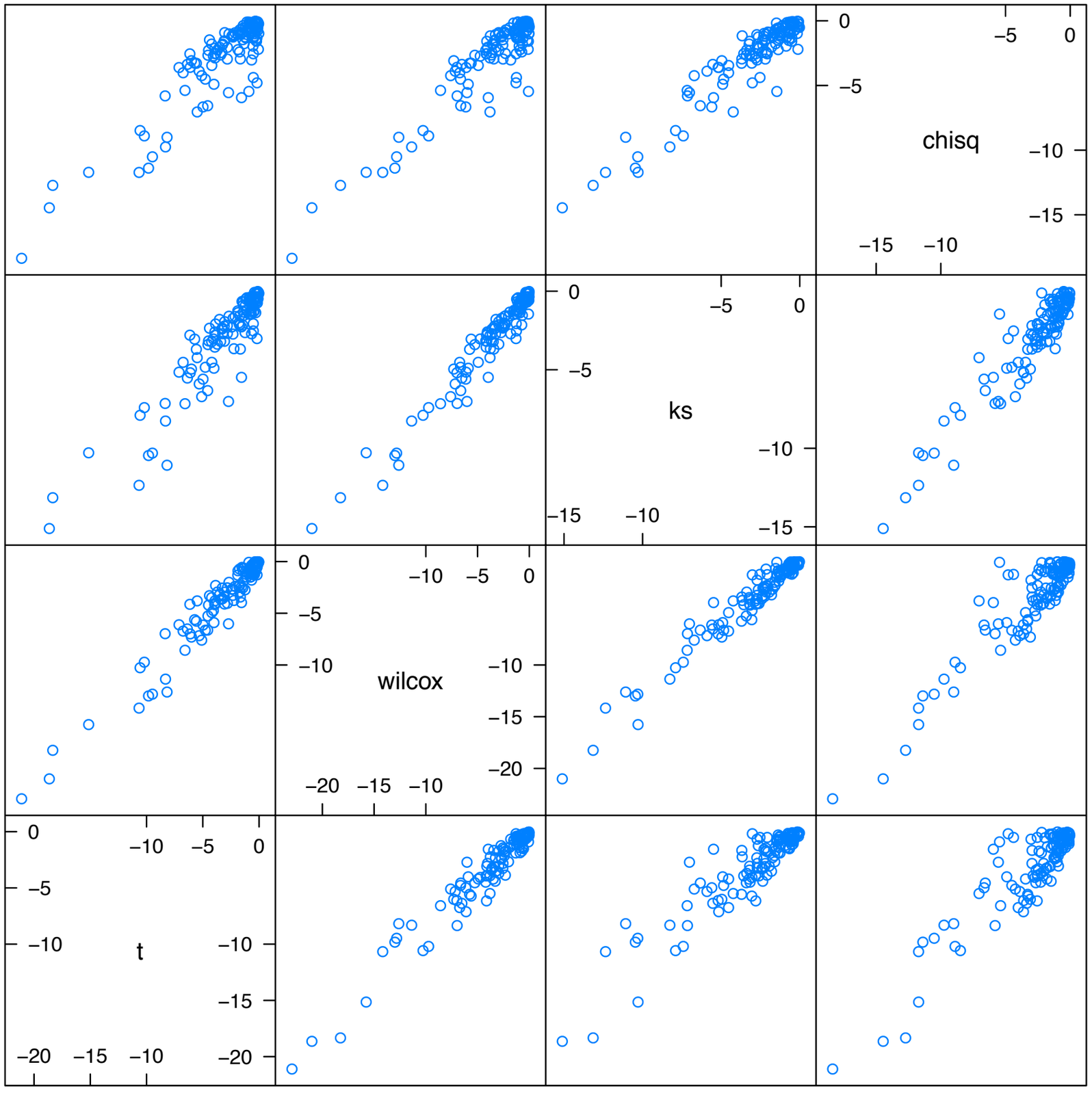
How many points will be in each panel?

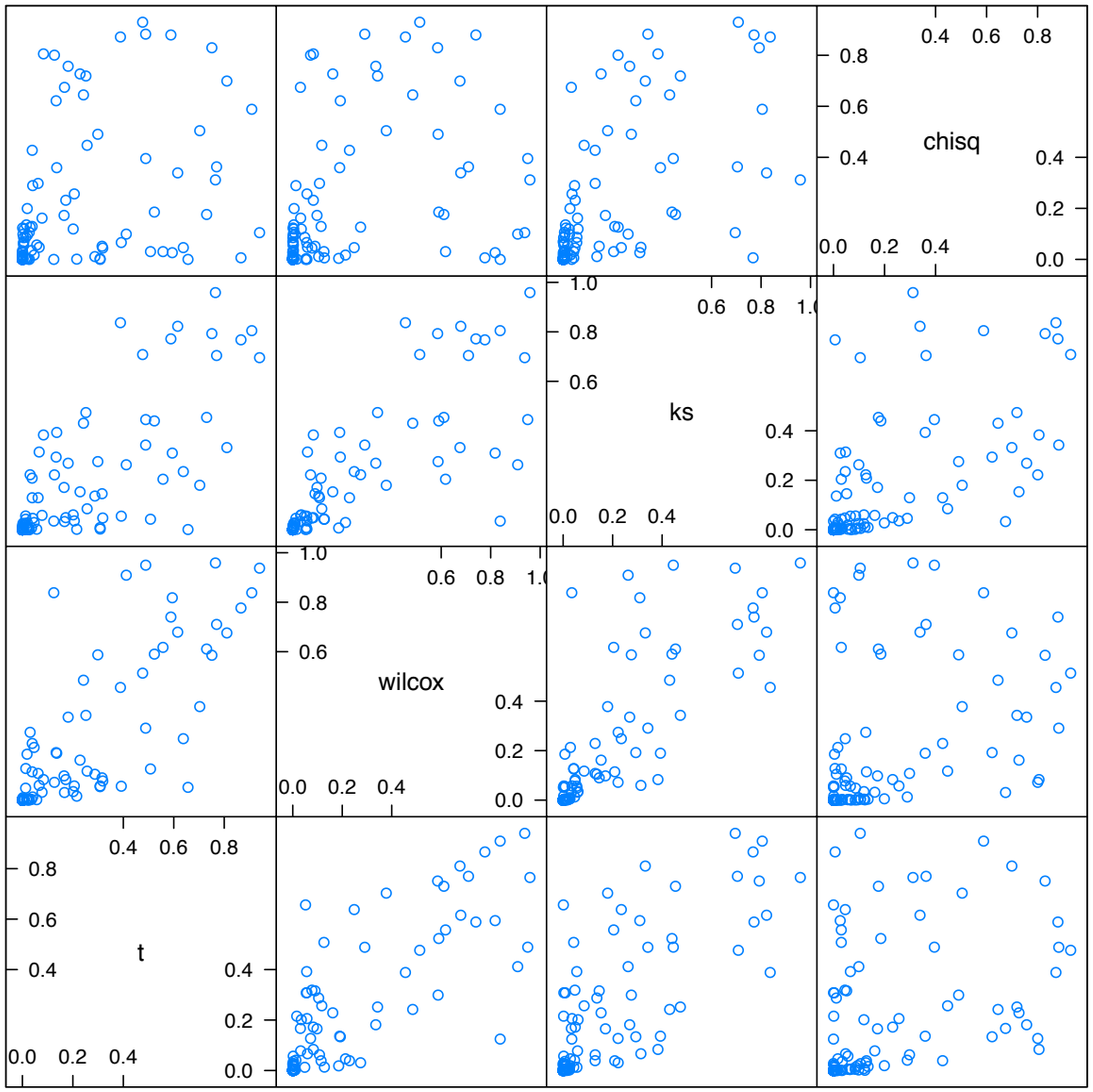
What do you expect these scatterplots to look like? Why?

```
> round(cor(subset(tgtRes, select = c(t, wilcox, ks, chisq))), 2)
          t wilcox   ks chisq
t      1.00    0.82 0.79  0.46
wilcox 0.82    1.00 0.83  0.47
ks     0.79    0.83 1.00  0.66
chisq  0.46    0.47 0.66  1.00
```

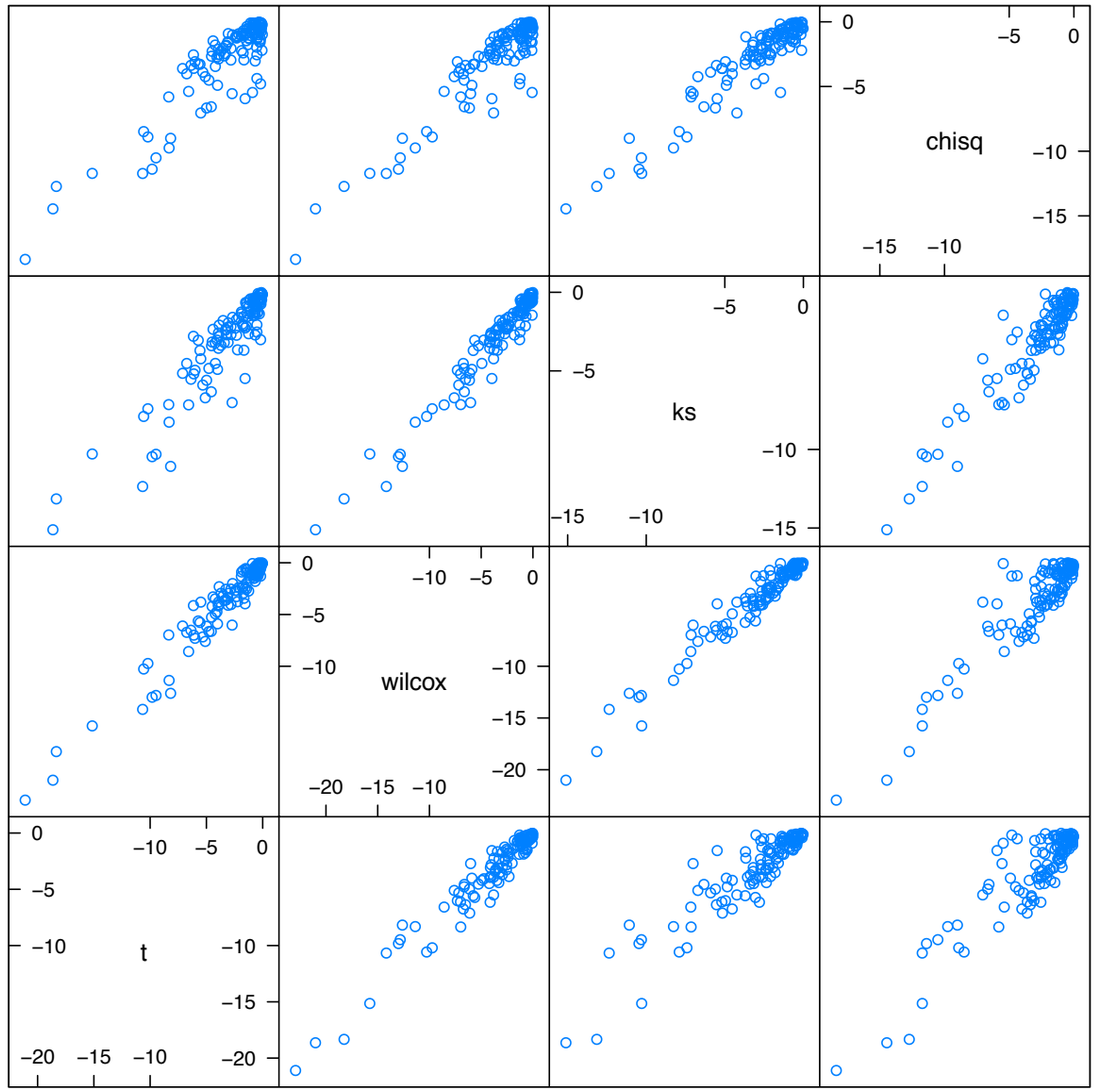


```
> round(cor(foo), 2)
          t wilcox   ks chisq
t      1.00   0.97 0.93  0.90
wilcox 0.97   1.00 0.98  0.93
ks     0.93   0.98 1.00  0.95
chisq  0.90   0.93 0.95  1.00
```

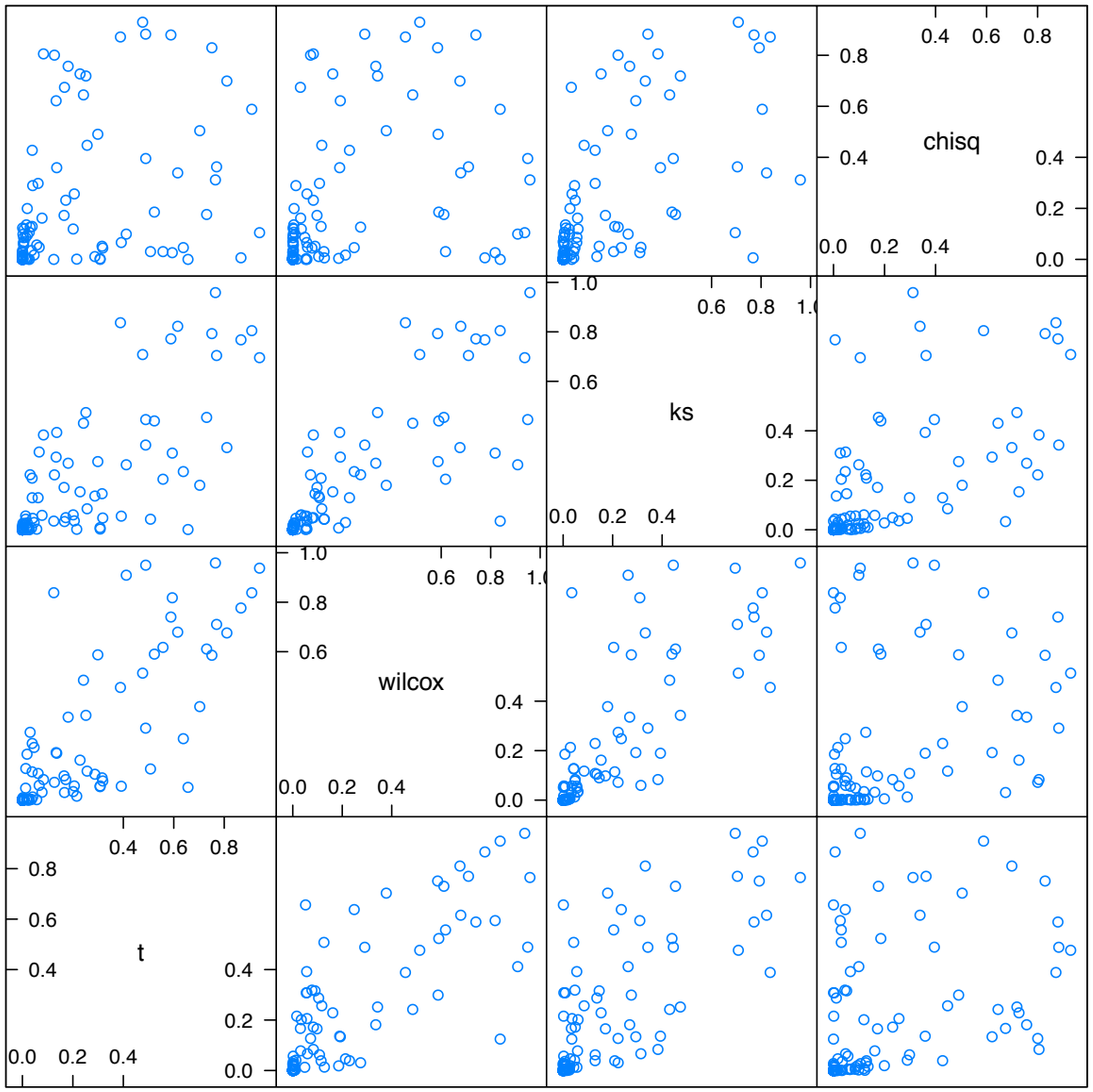


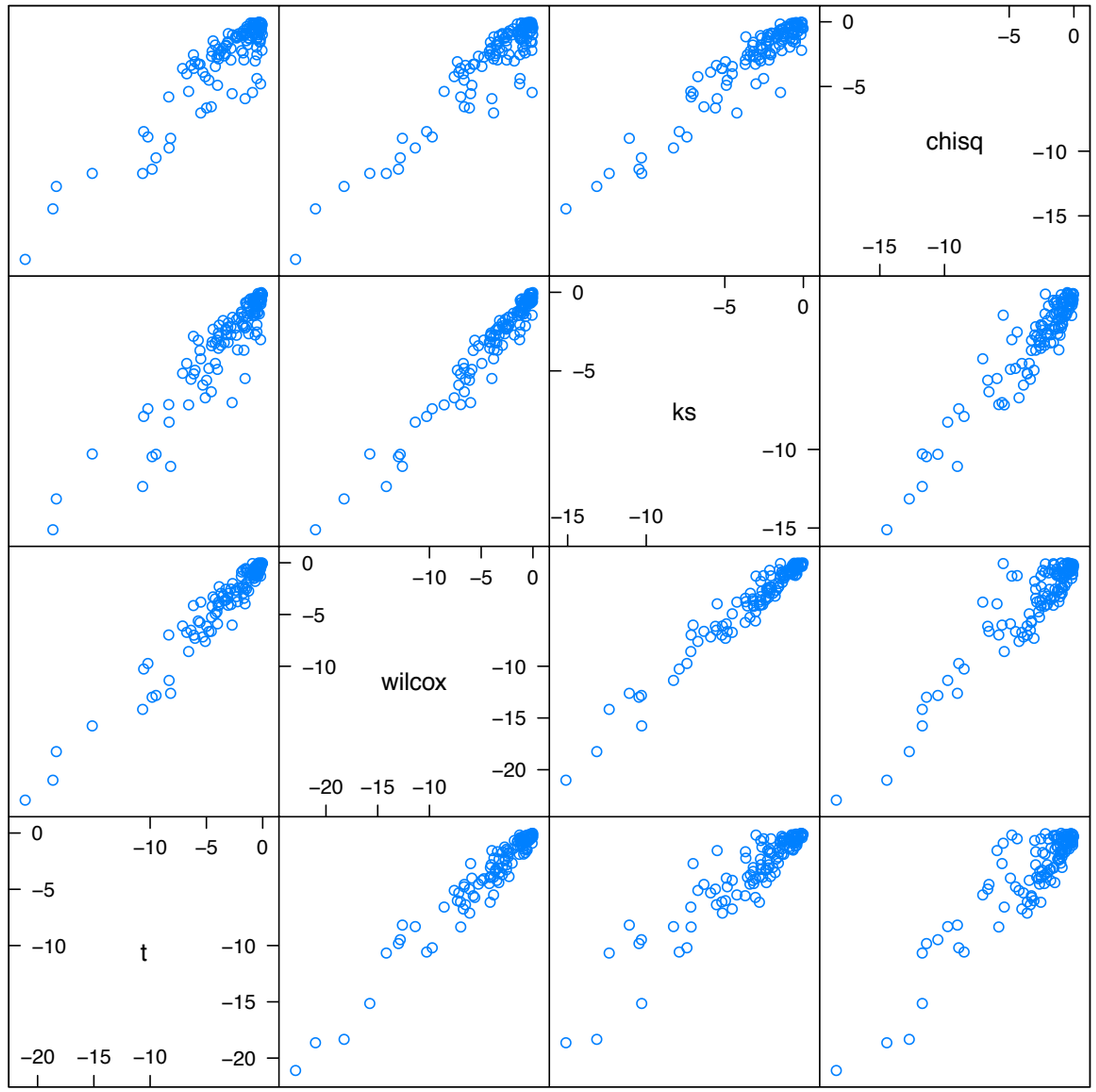


p−values from two groups tests



log10 p−values from two groups tests





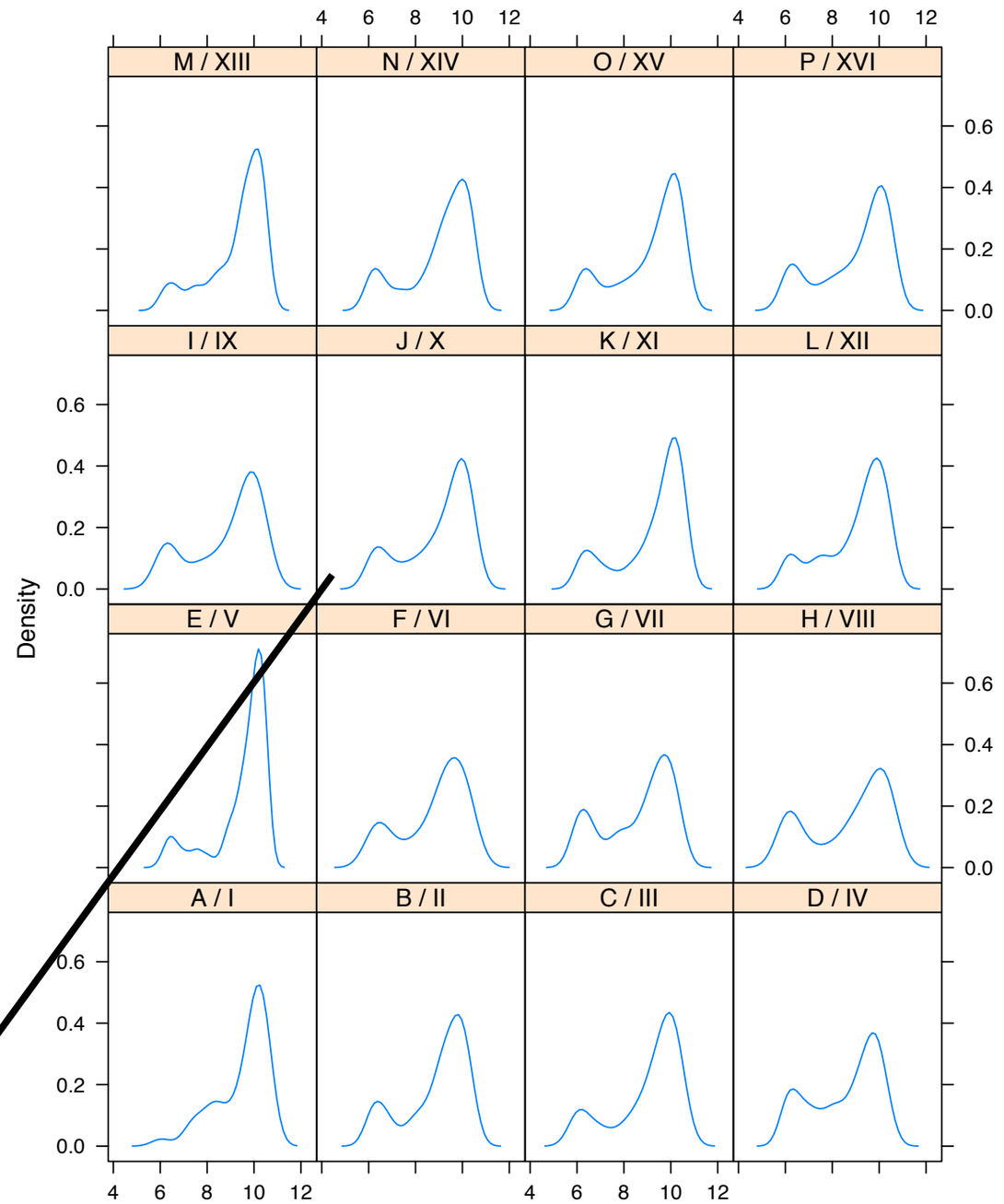
120 points in each panel, one per chromosome pair = two group comparison

strong agreement in ‘statistically significant’ region, elsewhere resembles indep. (?) U[0,1]

Now want to drill down to specific chromosome pairs.

Which ones are "different"?

Let's try to avoid the dreaded huge table of numbers .....

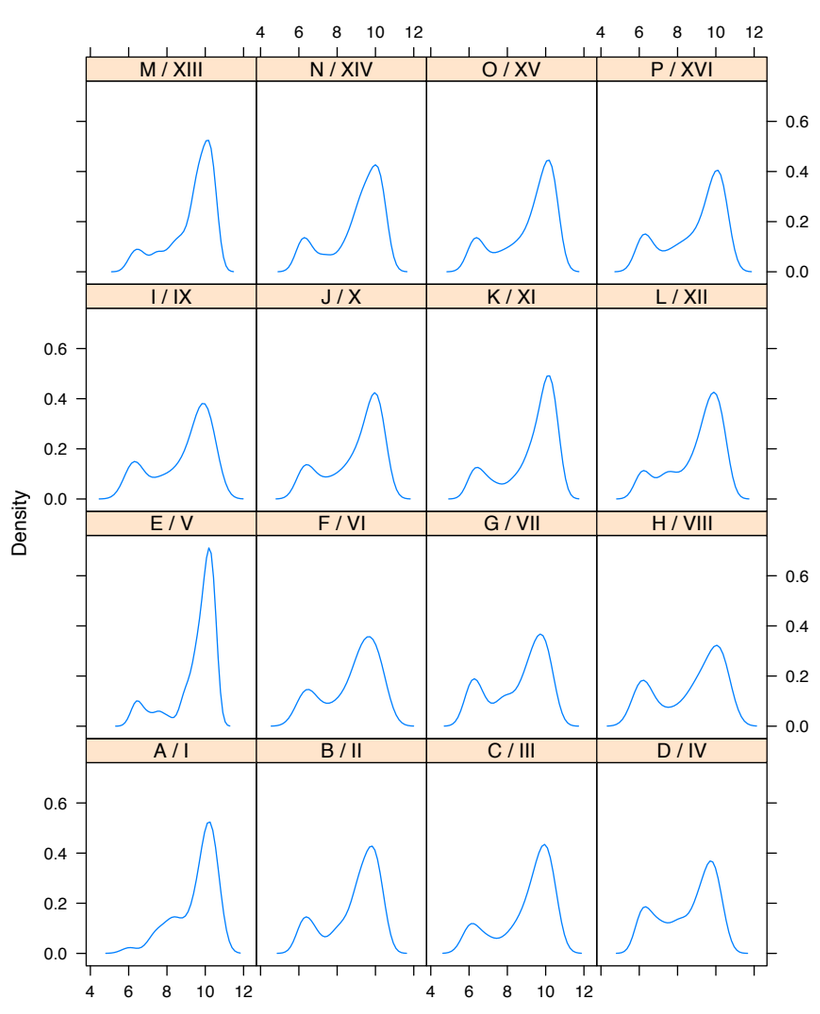


```
> head(tgtRes)
  chromoA chromoB             t         wilcox             ks         chisq
1       1       2 1.038025e-05 7.290265e-07 2.526472e-06 2.189111e-07
2       1       3 4.863497e-03 2.096621e-03 1.869599e-02 5.551262e-02
3       1       4 2.667323e-11 5.387564e-11 1.258664e-08 3.274657e-09
4       1       5 8.663625e-01 7.766479e-01 7.677386e-01 6.262497e-03
5       1       6 6.258230e-05 1.170805e-05 2.981307e-05 3.542402e-04
6       1       7 6.327104e-11 1.857242e-10 3.887342e-08 1.262049e-09
```

Consider one particular test, e.g. the Wilcoxon test, and focus just on those p-values.

Let's make a heatmap of these p-values.

First, reshape the p-value data.




```
> str(jP$wilcox)
 num [1:16, 1:16] 1.00 7.29e-07 2.10e-03 5.39e-11 7.77e-01 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:16] "A / I" "B / II" "C / III" "D / IV" ...
  ..$ : chr [1:16] "A / I" "B / II" "C / III" "D / IV" ...
> jP$wilcox[1:5, 1:5]
              A / I       B / II      C / III       D / IV        E / V
A / I   1.000000e+00 7.290265e-07 2.096621e-03 5.387564e-11 7.766479e-01
B / II  7.290265e-07 1.000000e+00 9.003812e-02 1.123049e-03 2.429079e-13
C / III 2.096621e-03 9.003812e-02 1.000000e+00 2.590718e-04 9.181813e-05
D / IV  5.387564e-11 1.123049e-03 2.590718e-04 1.000000e+00 1.154823e-23
E / V   7.766479e-01 2.429079e-13 9.181813e-05 1.154823e-23 1.000000e+00
```

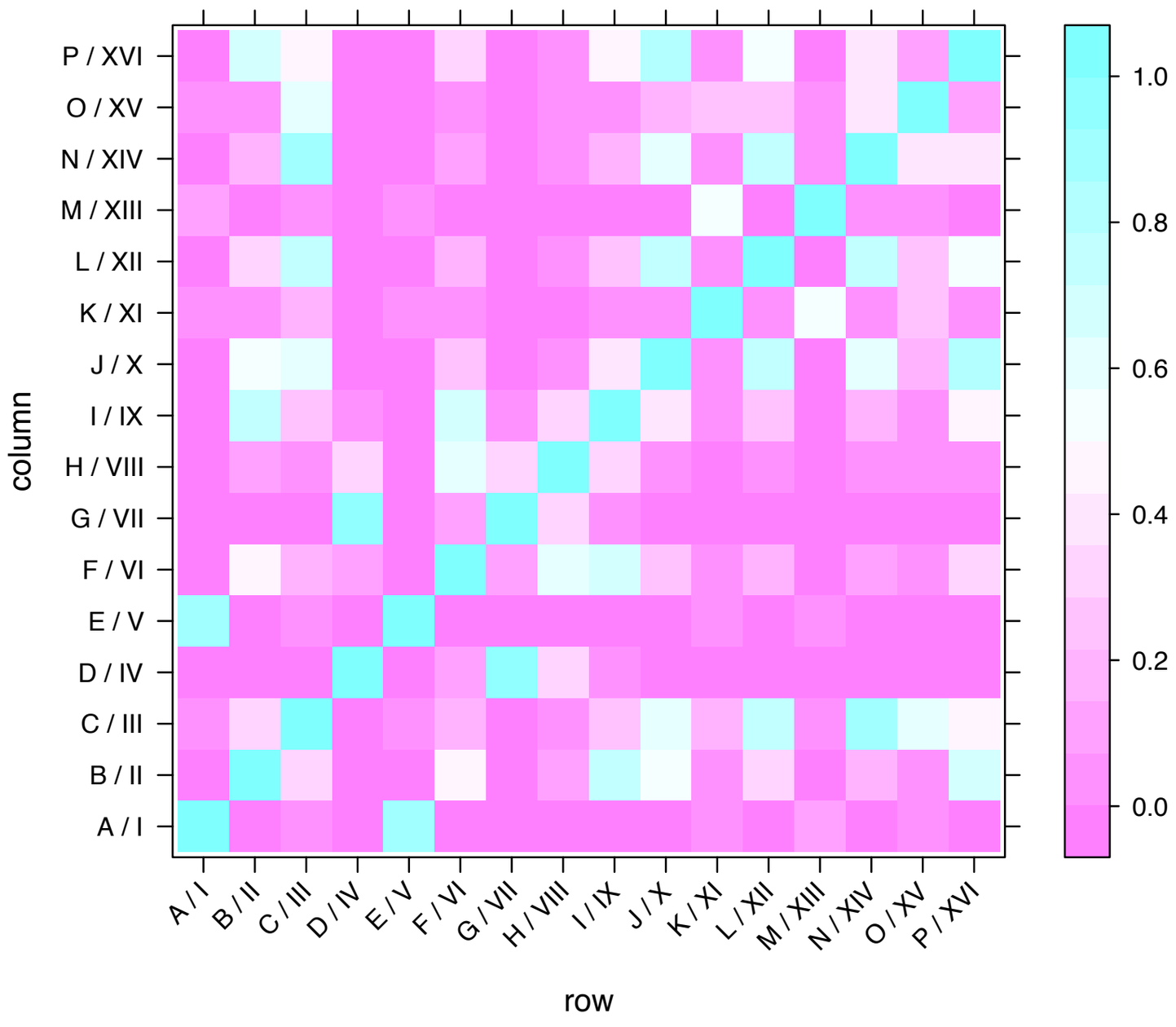
jP$wilcox is a 16 x 16 symmetric matrix of p-values.

Trivariate displays in lattice

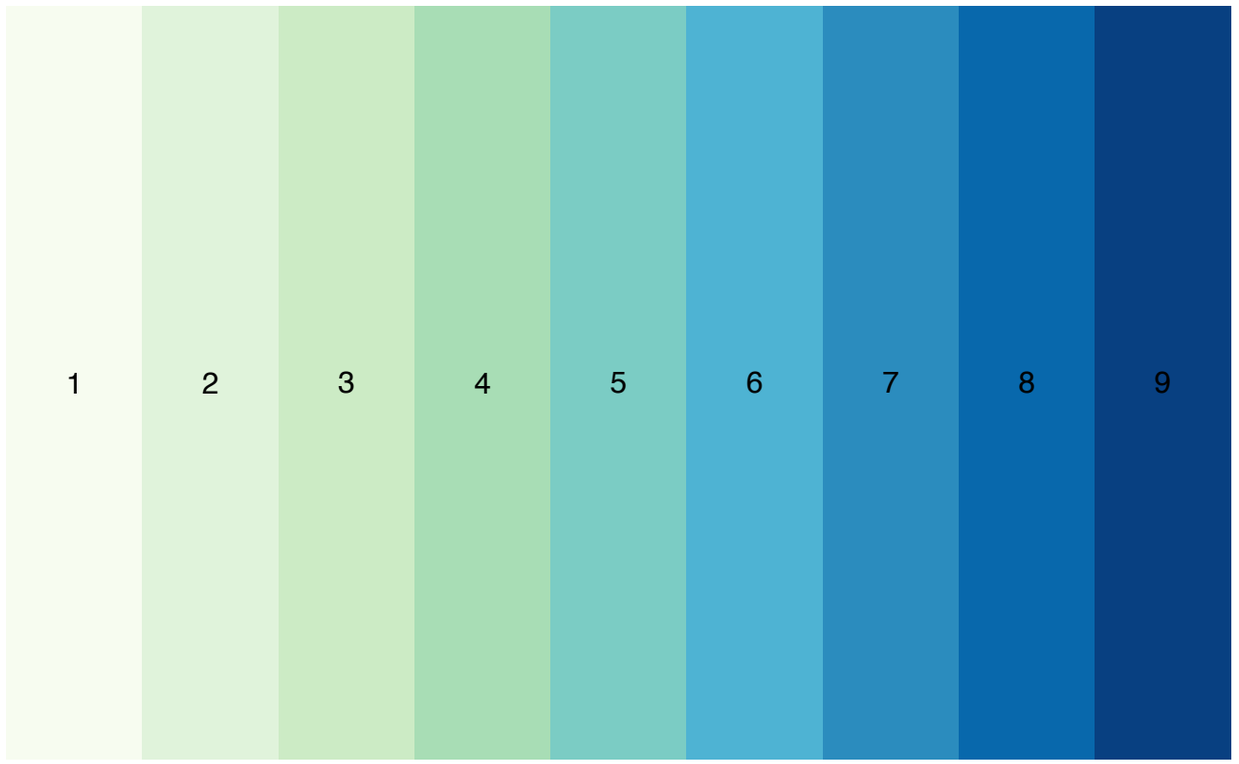
See Ch. 6 'Trivariate Displays' in Sarkar (2008).

- cloud() gives 3D scatterplots
- wireframe(), contourplot(), and levelplot() deal with surfaces and two-way tables
- Our matrix of p-values is a two-way table
  - Envision a square divided in to 16 * 16 cells
  - Each cell addresses a specific pair of chromosomes
  - Color the cell according to the evidence for a difference in distribution

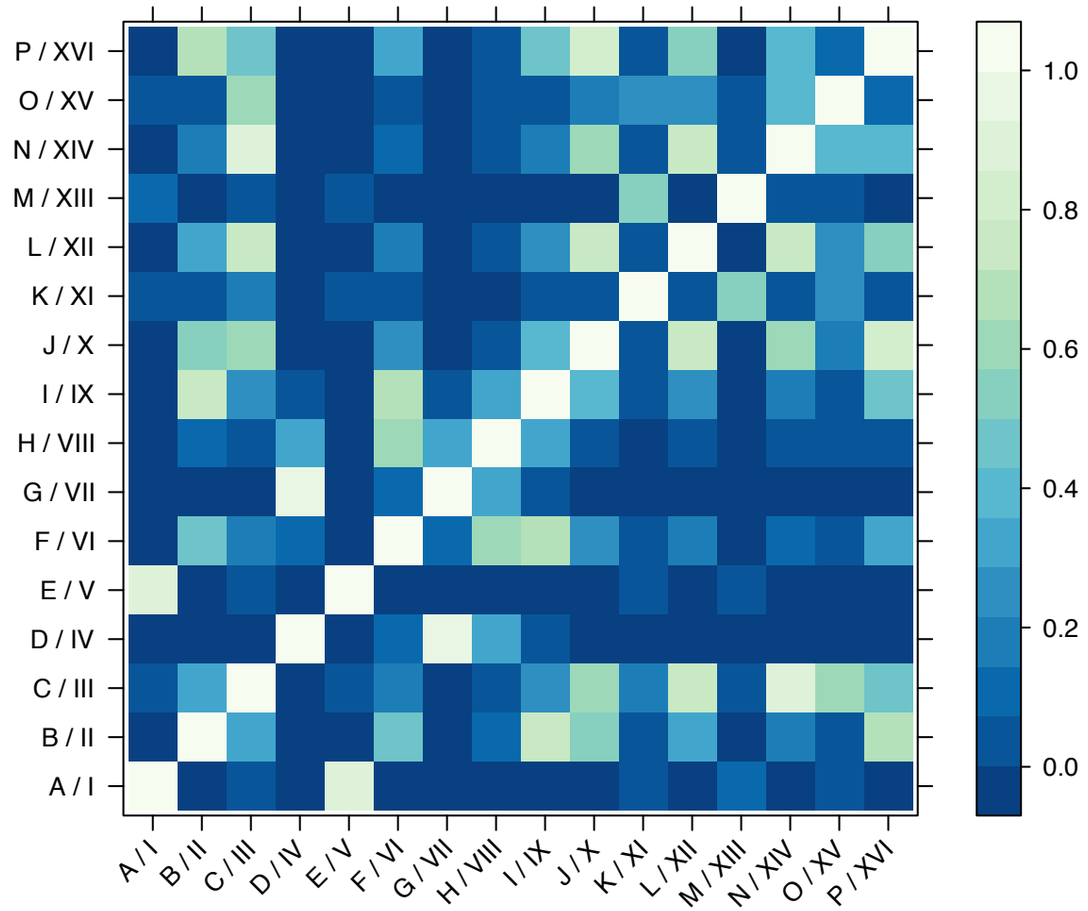
Warning: put your sunglasses on!




```
> ## graphs depicting the pairwise comparisons of phenotype dist'n
> ## shade the squares according to p-value
> levelplot(jP$t, scales = list(x = list(rot = 45)))   # wow, that's ugly
```

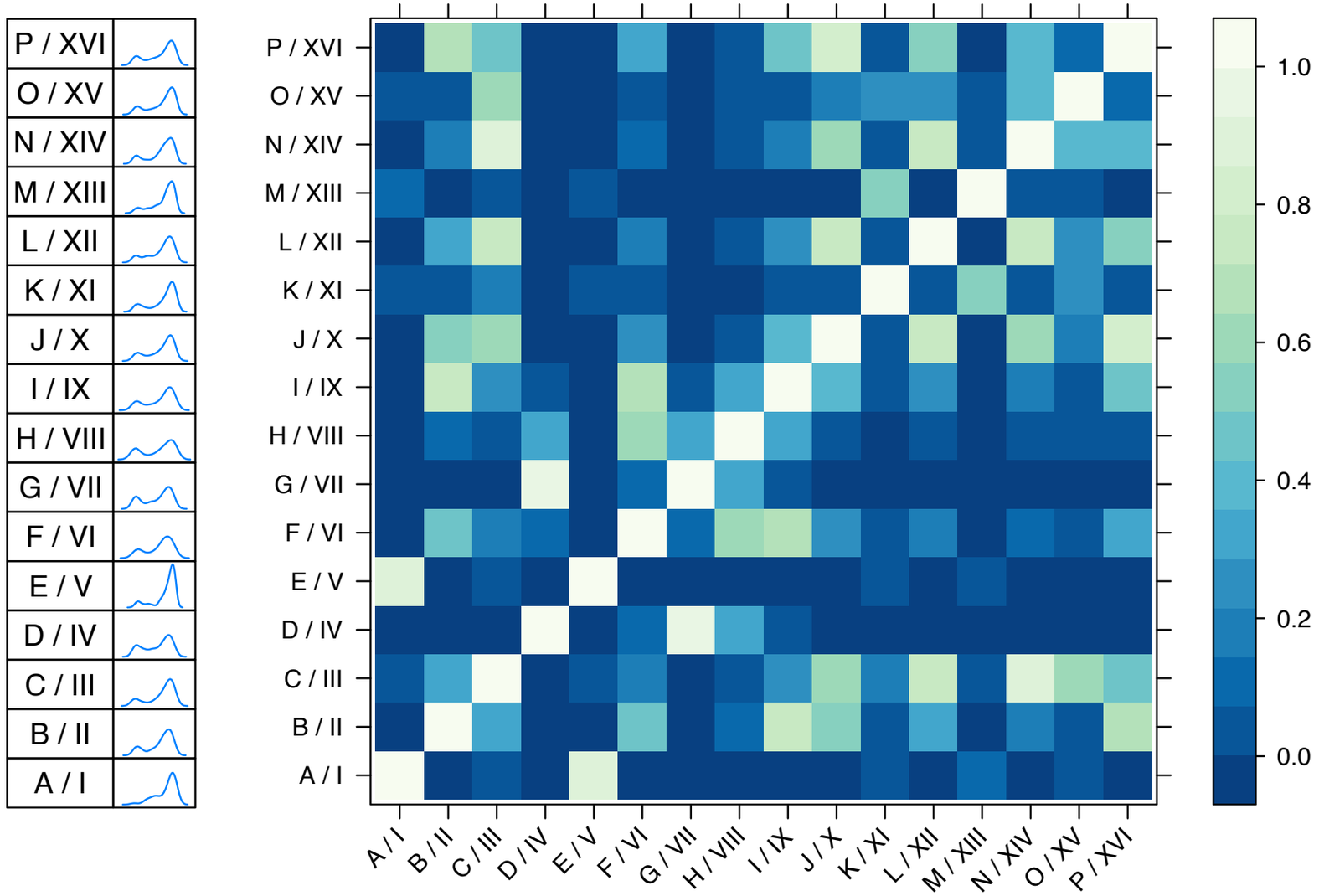



```
> display.brewer.pal(n = 9, "GnBu")
> text(x = 1:9, y = rep(1, 9), labels = 1:9)
> title("RColorBrewer palette 'GnBu'")
```



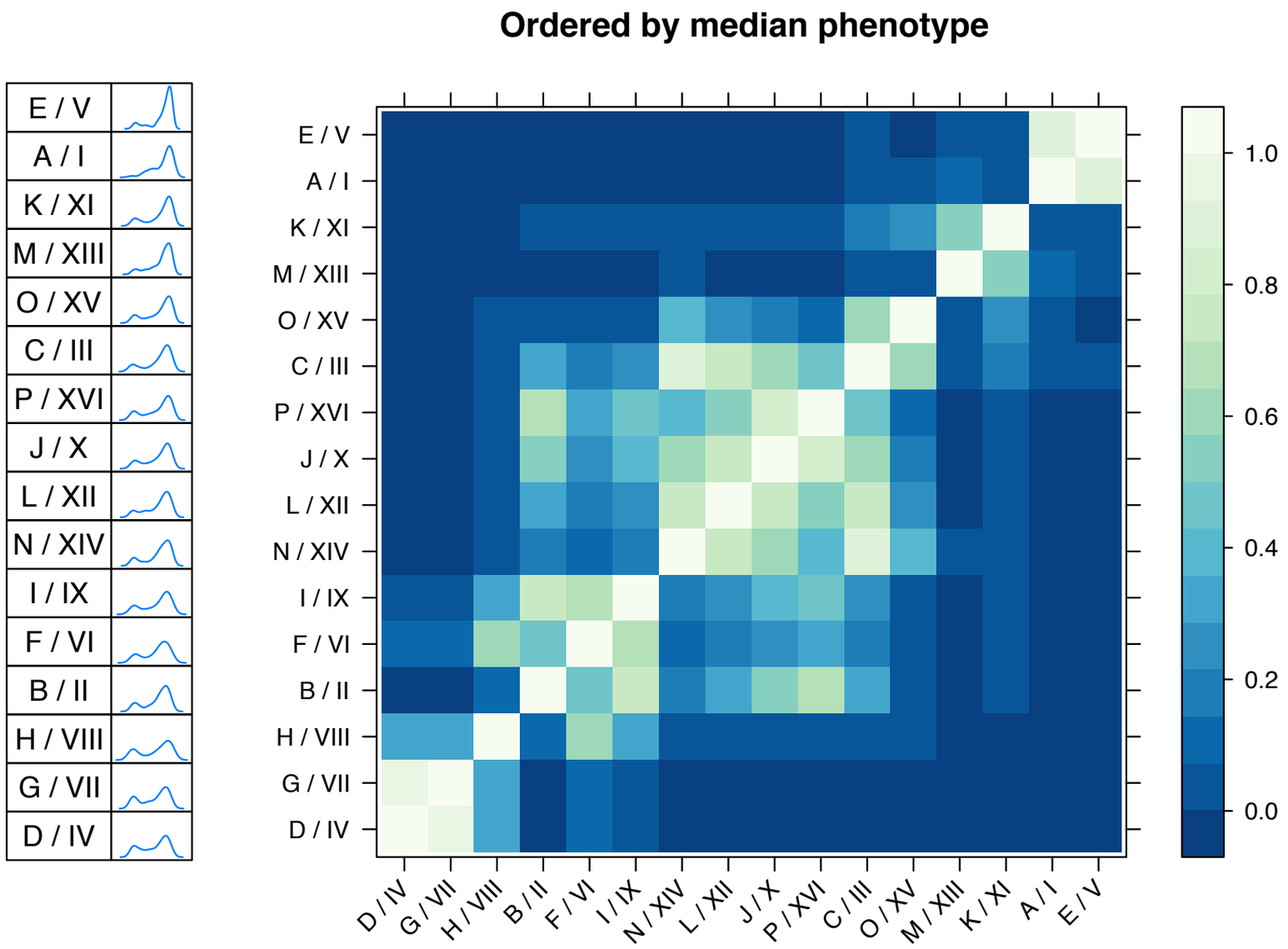

```
> ## pick palette to focus on
> brewPal <- brewer.pal(n=9,"GnBu")
> colorFun <- colorRampPalette(rev(brewPal))
> myCols <- colorFun(100)

> ## improved plot, color-wise
> levelplot(jP$t, col.regions = myCols, scales = list(x = list(rot = 45)),
+           xlab = "", ylab = "",
+           main = "Colors based on 'GnBu' palette from RColorBrewer")
```

Key idea: associate a p-value of 1 with white and a p-value of 0 with an actual color.



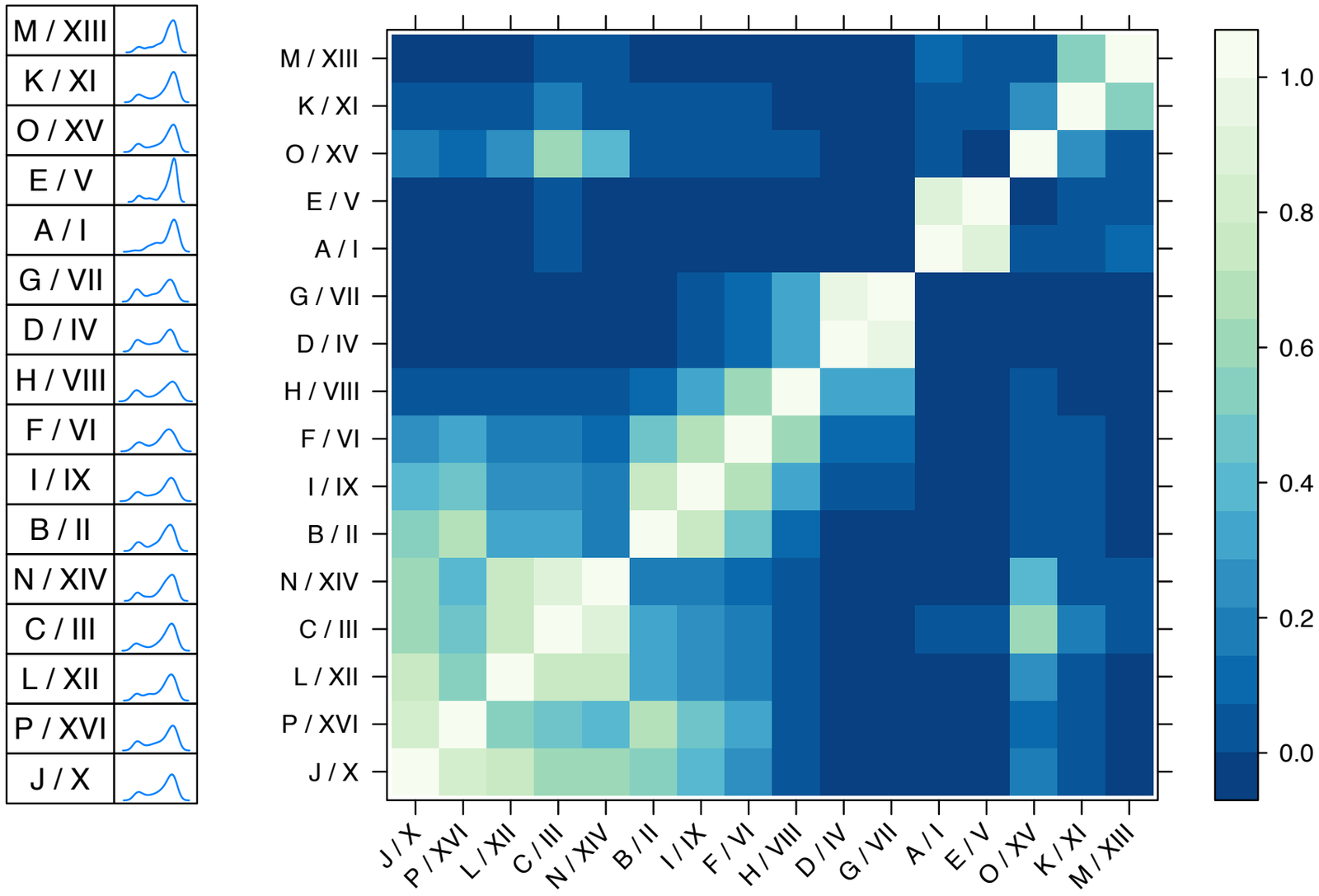

```
## including densityplots alongside is helpful
lPlot <- levelplot(jP$t, col.regions = myCols, xlab = "", ylab = "",
                   scales = list(x = list(rot = 45)))

dPlot <-
  densityplot(~ pheno | chromoPretty, hDat,
              layout = c(1, nC), plot.points = FALSE,
              strip = FALSE,
              strip.left = strip.custom(horizontal = TRUE,
                bg = NA),
              par.strip.text = list(lines = 3),
              scales = list(draw = FALSE),
              xlab = "", ylab = "")

print(dPlot, pos = c(0, 0.1, 0.2, 0.96), more = TRUE)
print(lPlot, pos = c(0.15, 0, 1, 1), more = FALSE)
```

Show data and modelling / inference results TOGETHER whenever you can.

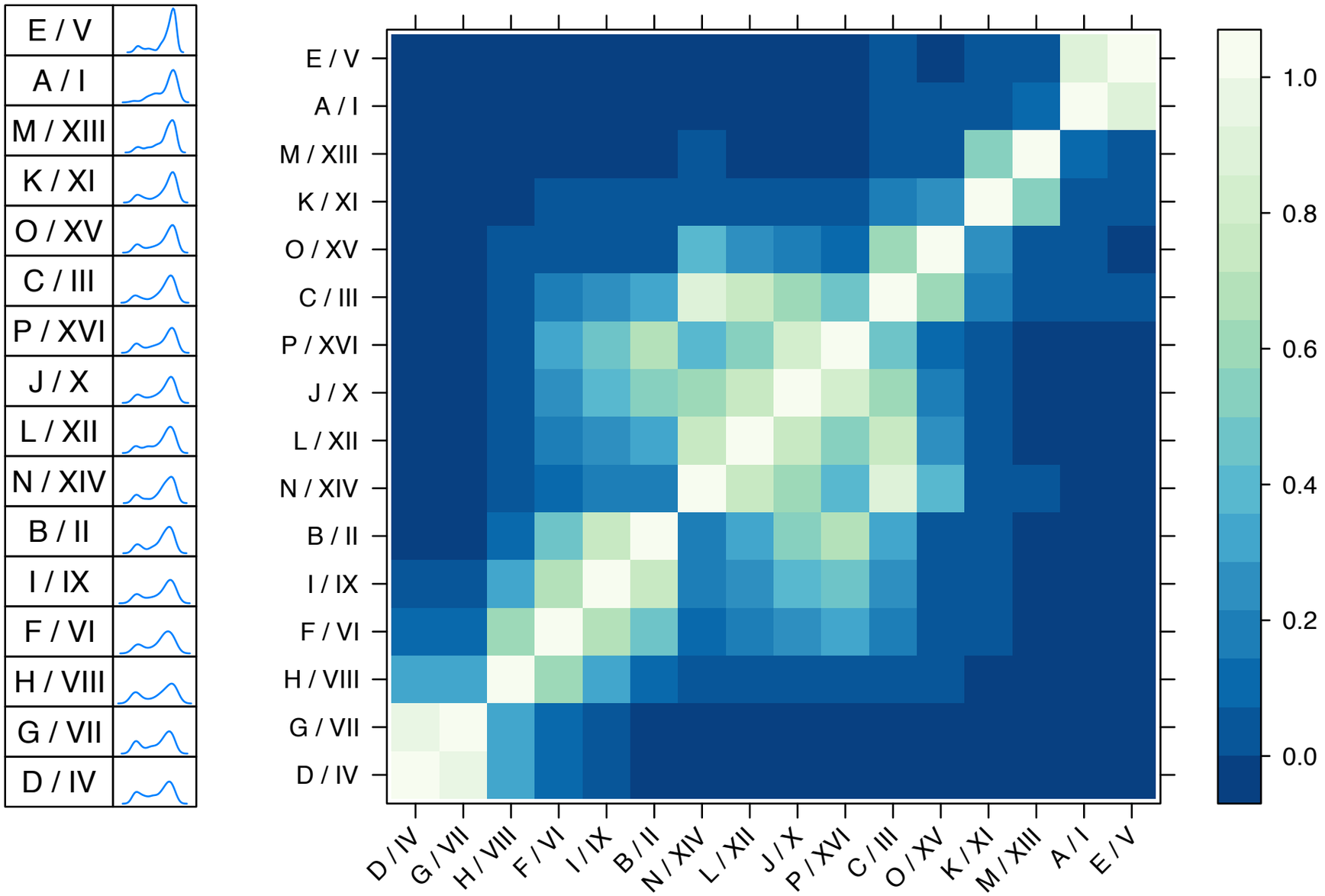



```
> jOrd1 <- order(tapply(hDat$pheno, hDat$chromoPretty, median))
> jTitle <- "Ordered by median phenotype"
> jOrd <- jOrd1
> lPlot <- levelplot(jP$t[jOrd, jOrd], col.regions = myCols,
+                    xlab = "", ylab = "", main = jTitle,
+                    scales = list(x = list(rot = 45)))
> dPlot <- update(dPlot, index.cond = list(jOrd))
> print(dPlot, pos = c(0, 0.1, 0.2, 0.96), more = TRUE)
> print(lPlot, pos = c(0.15, 0, 1, 1), more = FALSE)
```

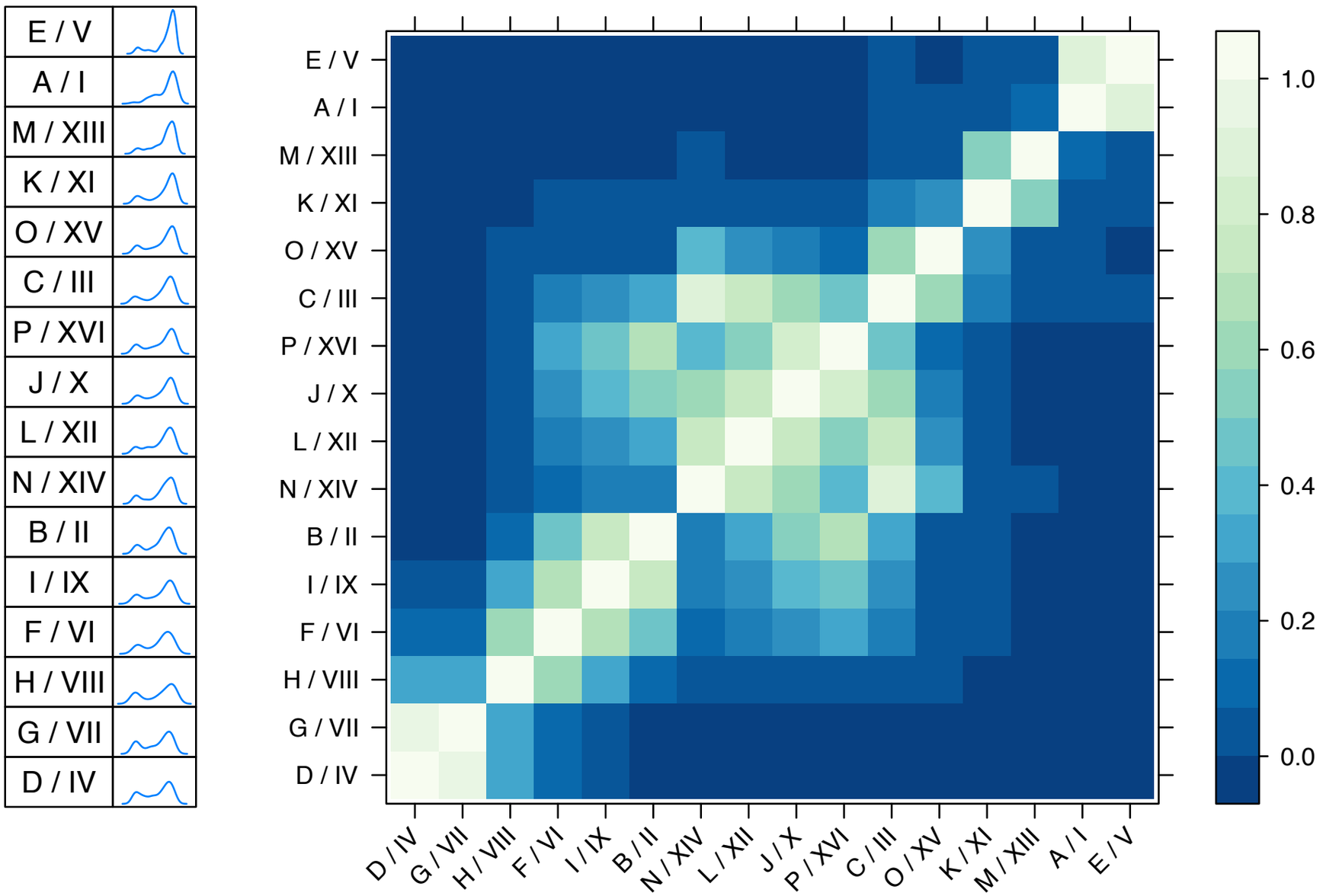
Use a rationale, deliberate ordering.




```
> jOrd2 <- order.dendrogram(as.dendrogram(hclust(dist(jP))))
> jTitle <- "Ordered by clustering result"
> jOrd <- jOrd2
> lPlot <- levelplot(jP$t[jOrd, jOrd], col.regions = myCols,
+                    xlab = "", ylab = "", main = jTitle,
+                    scales = list(x = list(rot = 45)))
> dPlot <- update(dPlot, index.cond = list(jOrd))
> print(dPlot, pos = c(0, 0.1, 0.2, 0.96), more = TRUE)
> print(lPlot, pos = c(0.15, 0, 1, 1), more = FALSE)
```



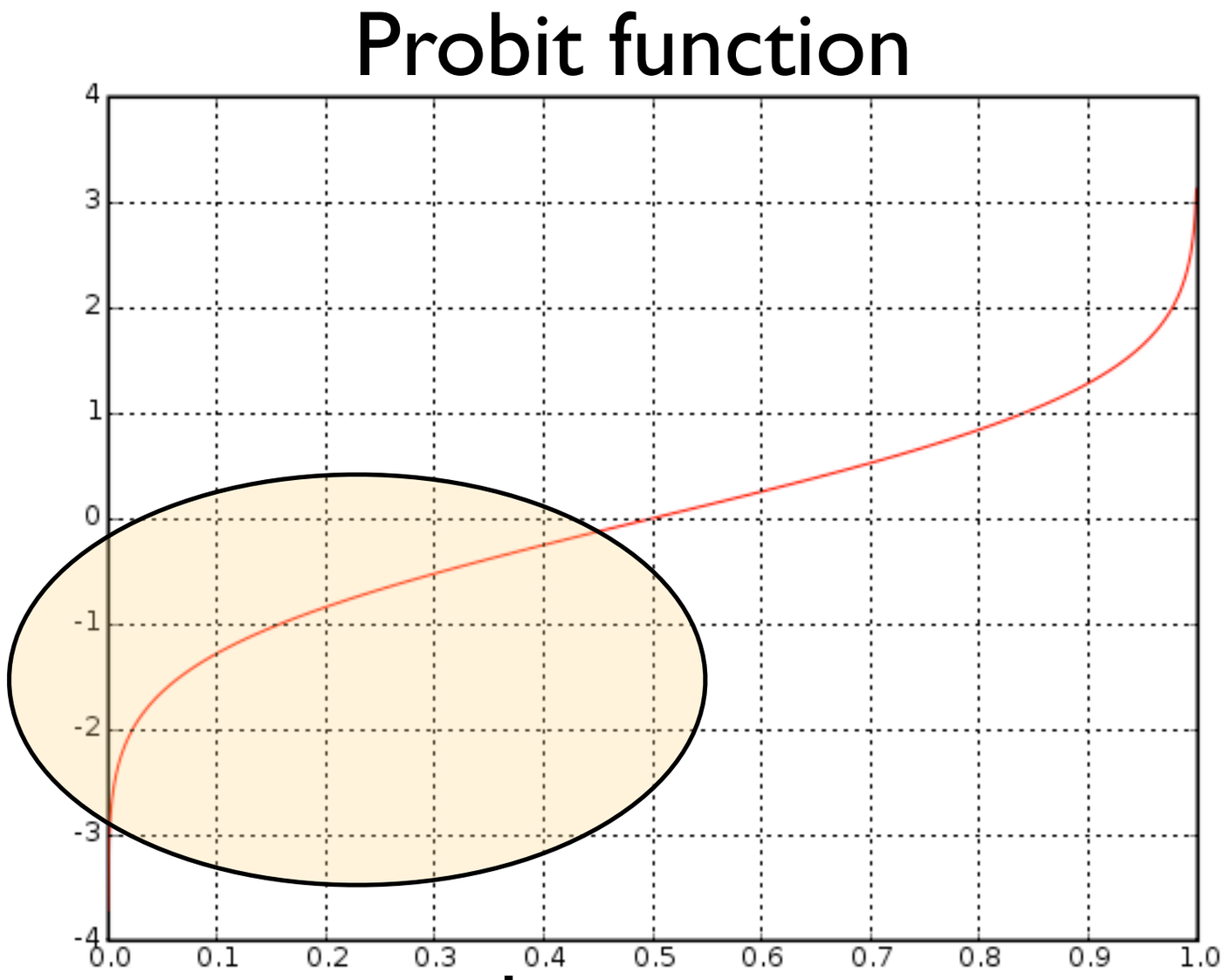

```
> jOrd3 <-
+   c(4, 7, 8, 6, 9, 2, 14, 12, 10, 16, 3, 15, 11, 13, 1,  5)
> jTitle <- "Ordered by Jenny"
> jOrd <- jOrd3
> lPlot <- levelplot(jP$t[jOrd, jOrd], col.regions = myCols,
+                    xlab = "", ylab = "", main = jTitle,
+                    scales = list(x = list(rot = 45)))
> dPlot <- update(dPlot, index.cond = list(jOrd))
> print(dPlot, pos = c(0, 0.1, 0.2, 0.96), more = TRUE)
> print(lPlot, pos = c(0.15, 0, 1, 1), more = FALSE)
```



Problem: we're using most of our color space to depict distinctions we don't care about, i.e. p-value = 0.3 vs. 0.8.

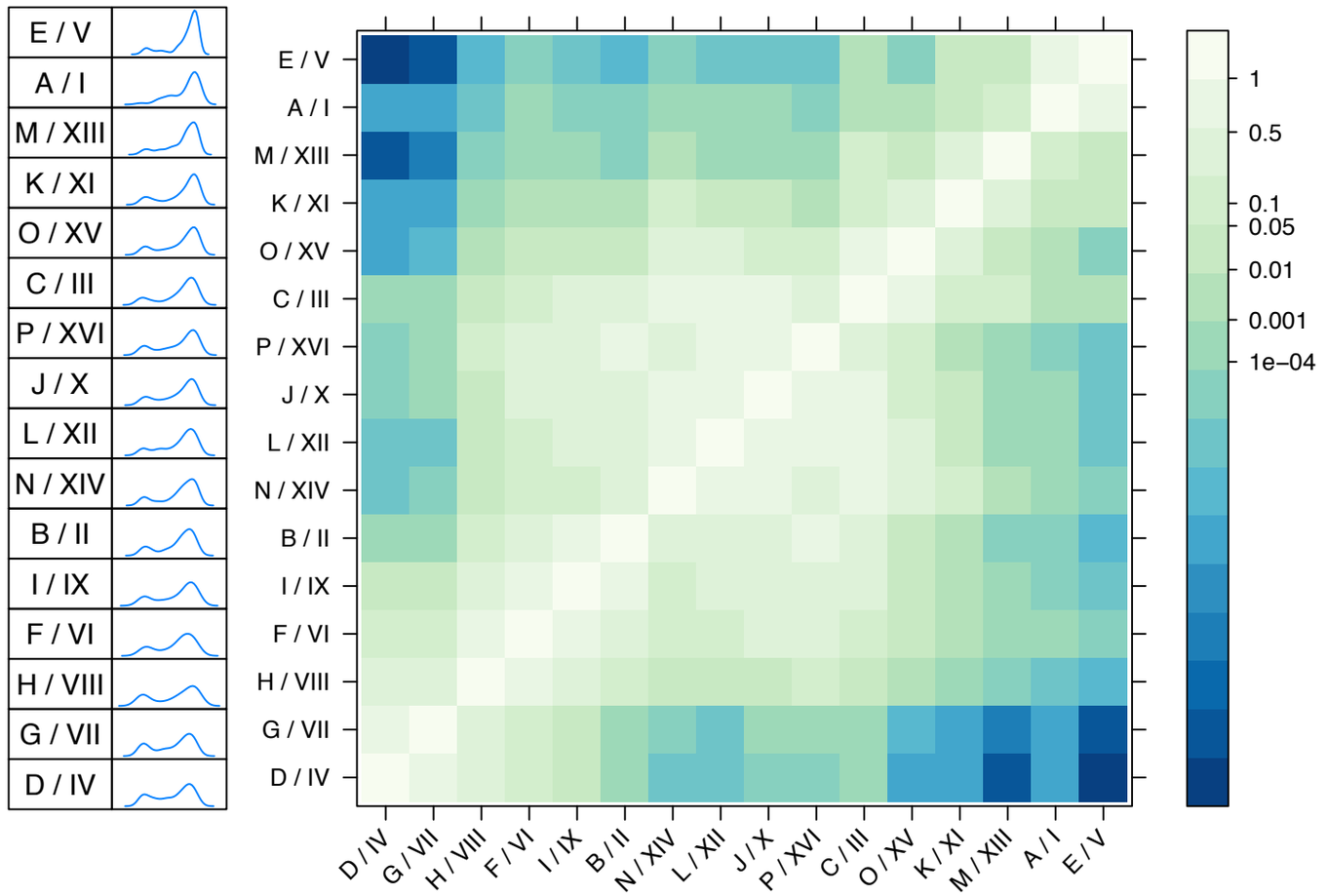
How we will use:

transformed p-values to drive the coloring

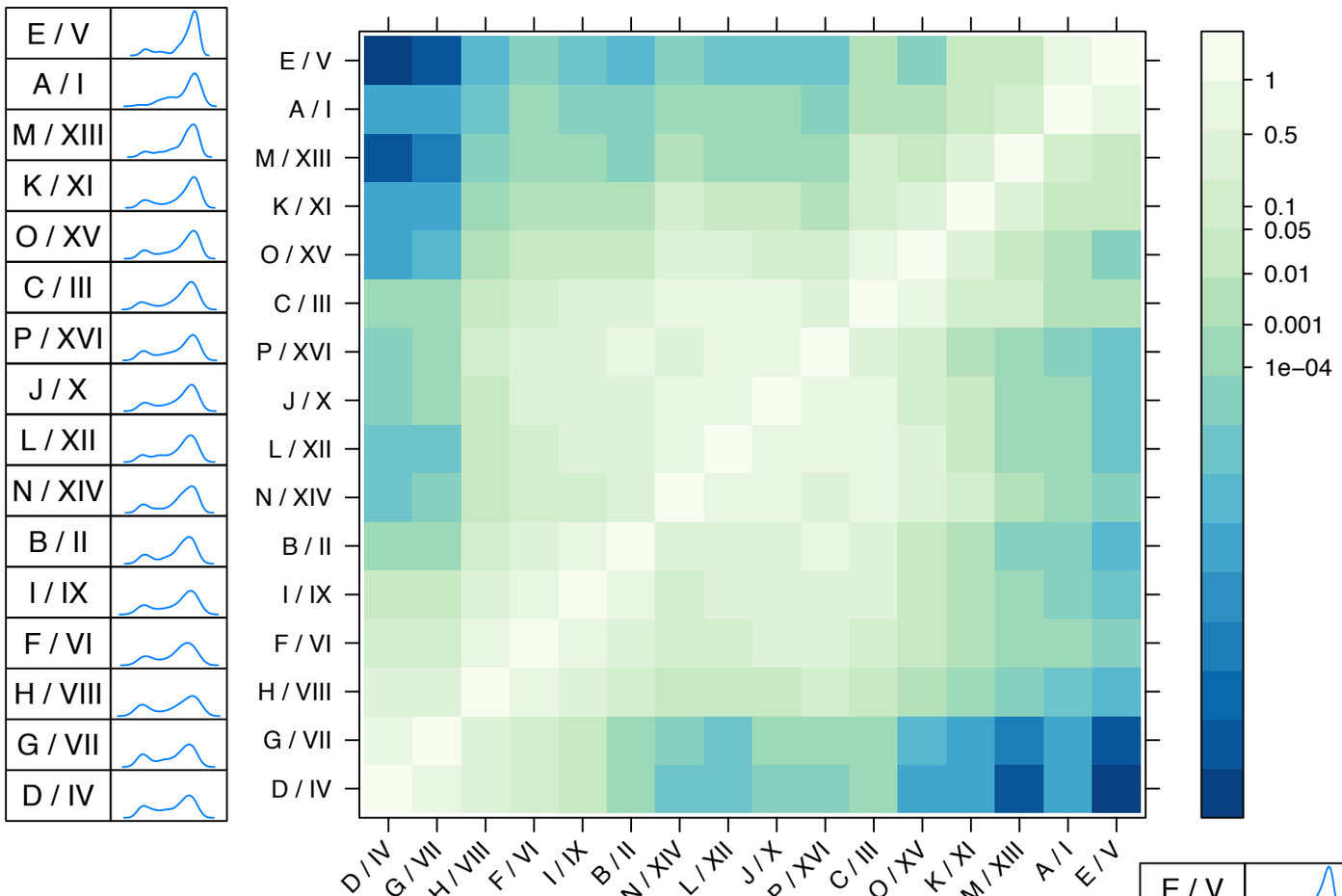


This transformation will help emphasize differences in small to modest p-values and de-emphasize difference in medium to large p-values.

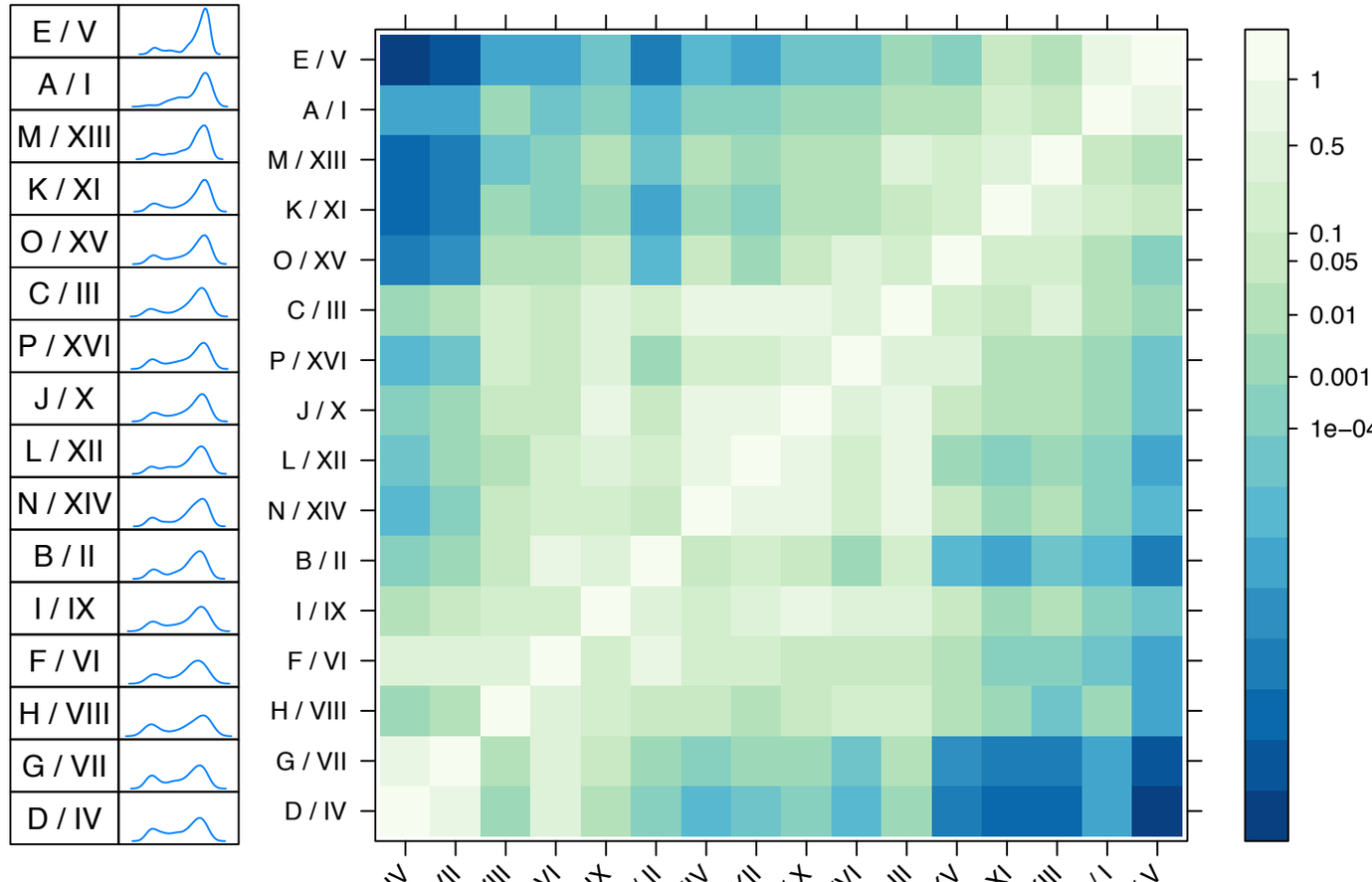
Figure from Wikipedia

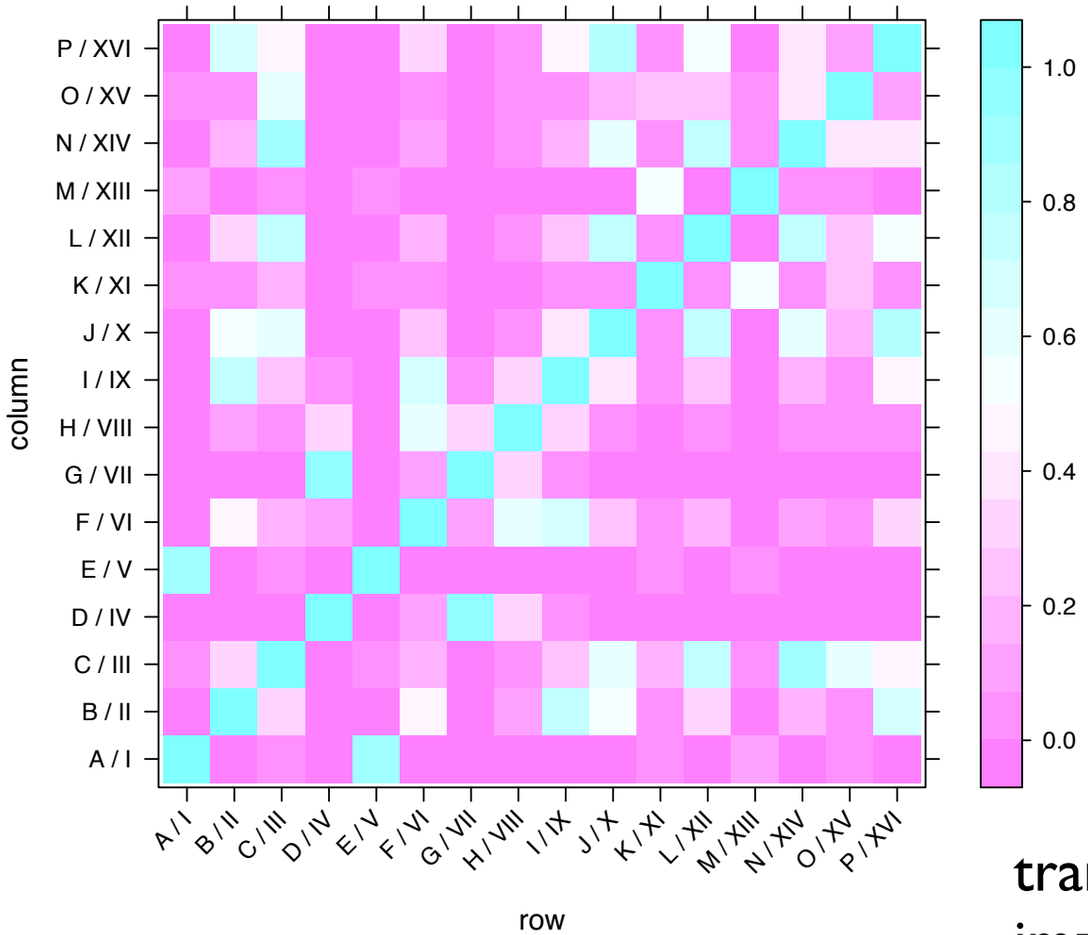



```
## transforming the p-values to optimize mapping into the color ramp
## I want to de-emphasize differences between teeny p-values
## and between very large p-values
## and increase the visual impact of differences in (0.01, 0.10)
pvalueTicks <- c(0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1)
qnScale <- 0.6
colorkeyTicks <- qnorm(pvalueTicks * qnScale)
jTitle <- "t-test, JB order, probit transformed p-values"

lPlot <-
  levelplot(qnorm$t(jP[jOrd, jOrd] * qnScale), col.regions = myCols,
            scales = list(x = list(rot = 45)),
            xlab = "", ylab = "", main = jTitle,
            colorkey = list(col = myCols,
              labels = list(labels = pvalueTicks, at = colorkeyTicks)))

print(dPlot, pos = c(0, 0.1, 0.2, 0.96), more = TRUE)
print(lPlot, pos = c(0.15, 0, 1, 1), more = FALSE)
```

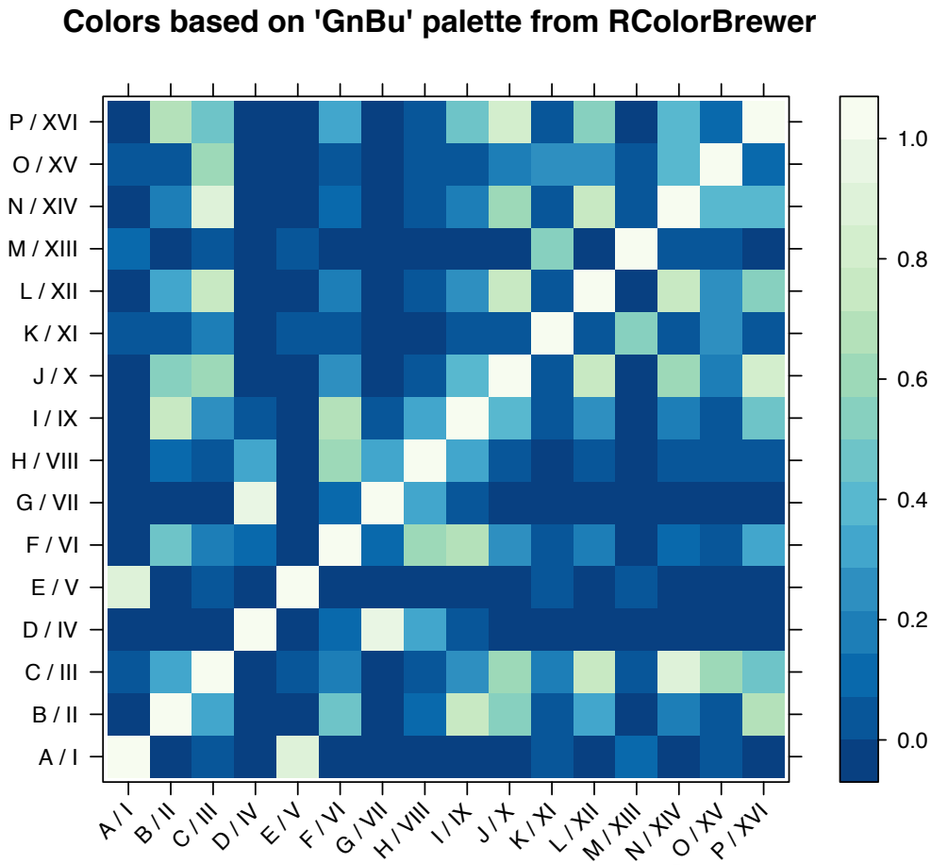


Results are similar, although there is ‘more statistical significance’ found by KS test here.



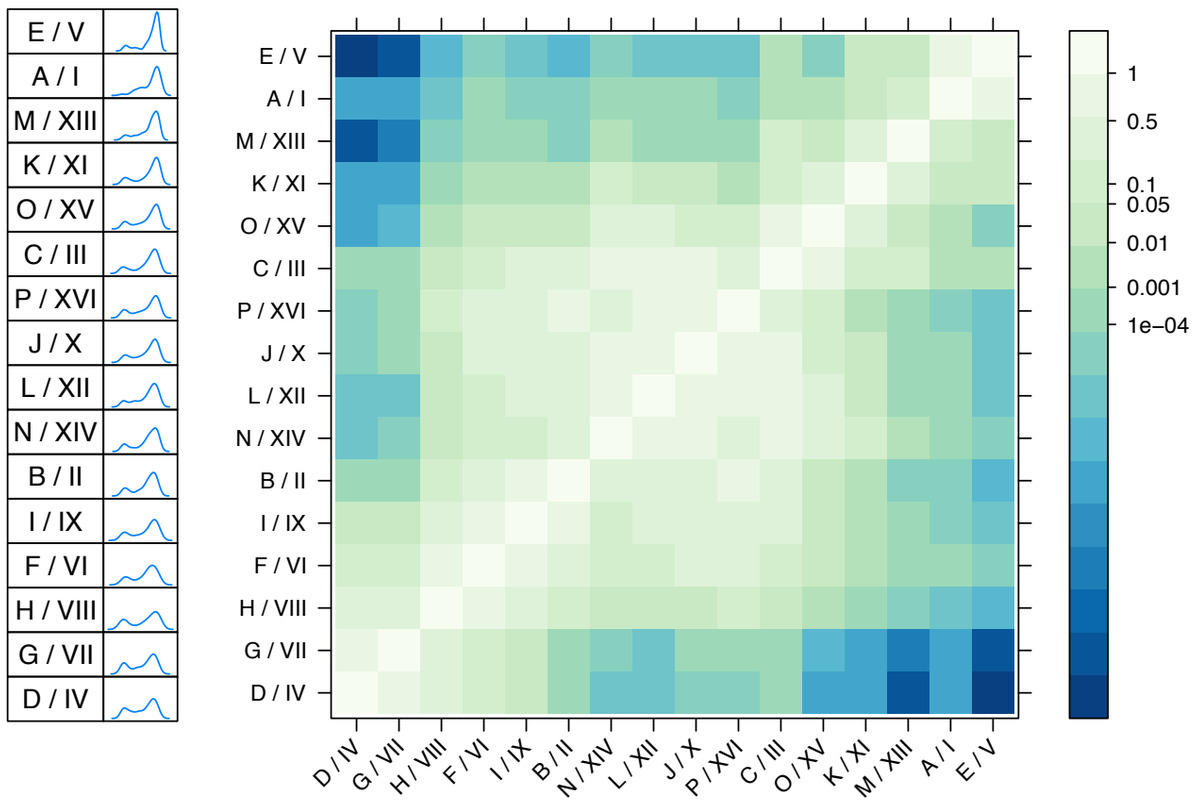


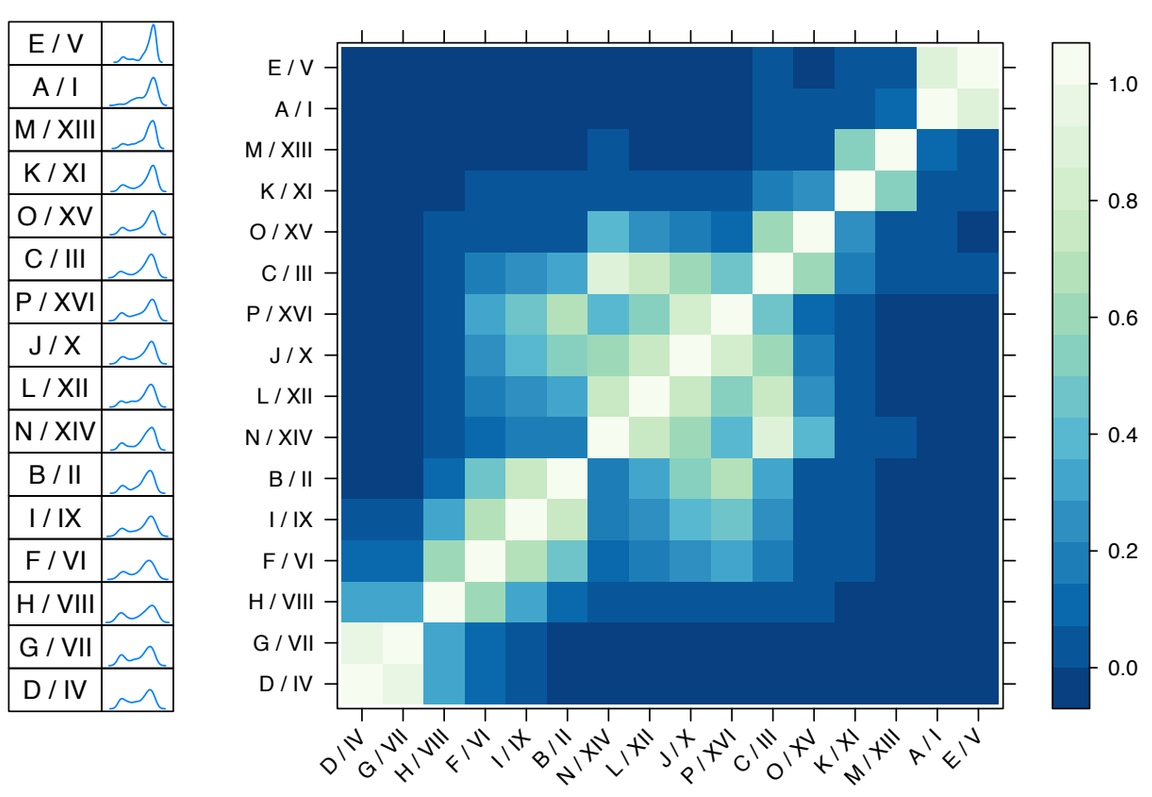
rational color scale

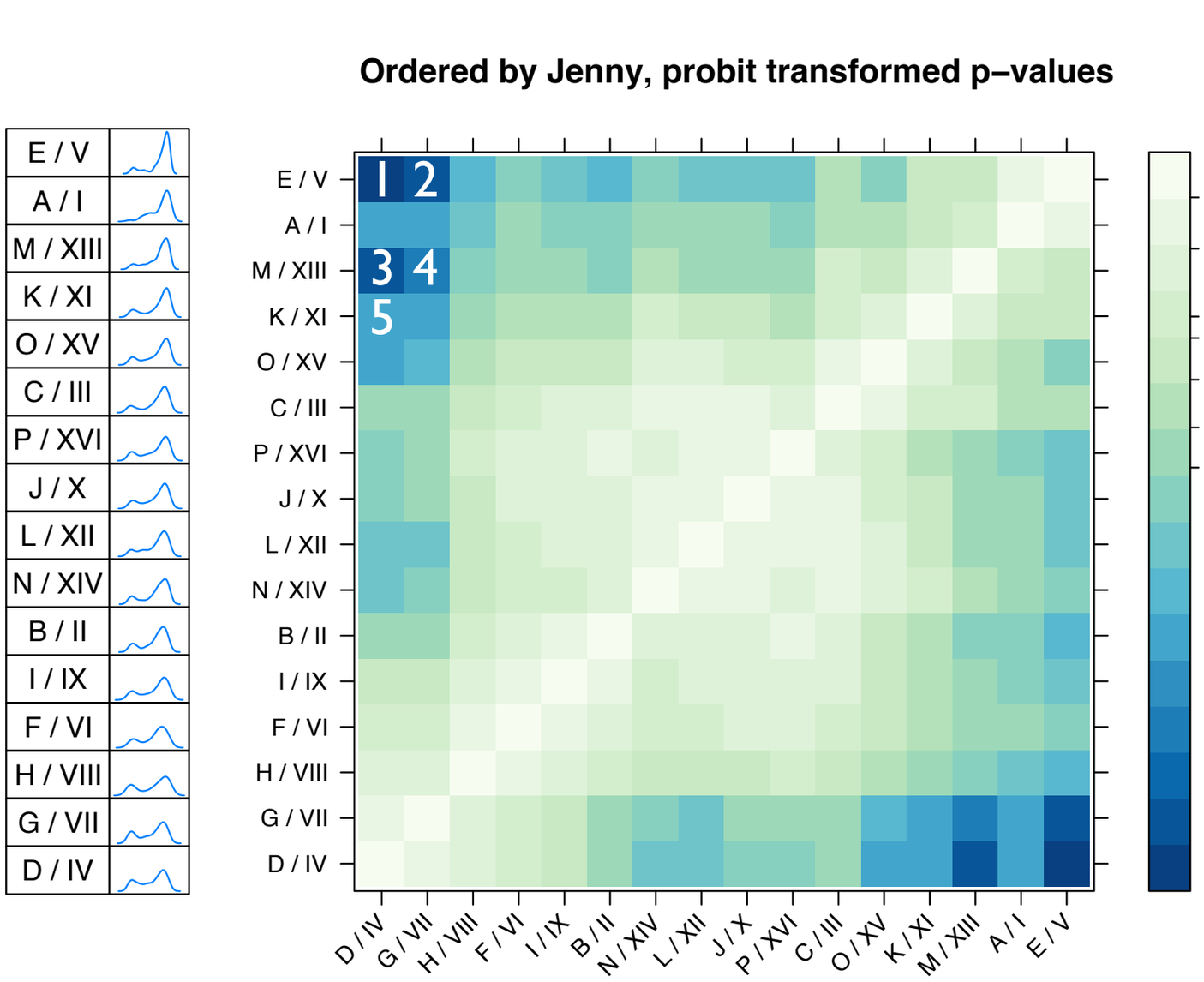


transformation emphasizes important differences

rational order, add densities








```
> head(tgtRes[order(tgtRes$t), ], 15)
    chromoA chromoB             t       wilcox
43        4       5 7.906832e-22 1.154823e-23 4.440
56        5       7 2.299423e-19 9.751576e-22 7.771
51        4      13 4.627917e-19 5.634388e-19 7.138
81        7      13 7.149913e-16 1.722451e-16 5.147
49        4      11 2.159276e-11 6.787809e-15 4.389
3         1       4 2.667323e-11 5.387564e-11 1.258
6         1       7 6.327104e-11 1.857242e-10 3.887
53        4      15 1.507109e-10 1.018281e-13 3.461
79        7      11 3.333448e-10 1.508345e-13 4.924
57        5       8 4.496093e-09 1.057575e-07 7.152
83        7      15 4.855781e-09 4.288748e-12 5.441
18        2       5 6.627043e-09 2.429079e-13 8.360
65        5      16 7.659286e-08 7.746156e-07 7.007
50        4      12 1.783292e-07 1.877395e-07 2.964
61        5      12 2.634741e-07 2.634340e-09 6.877
```

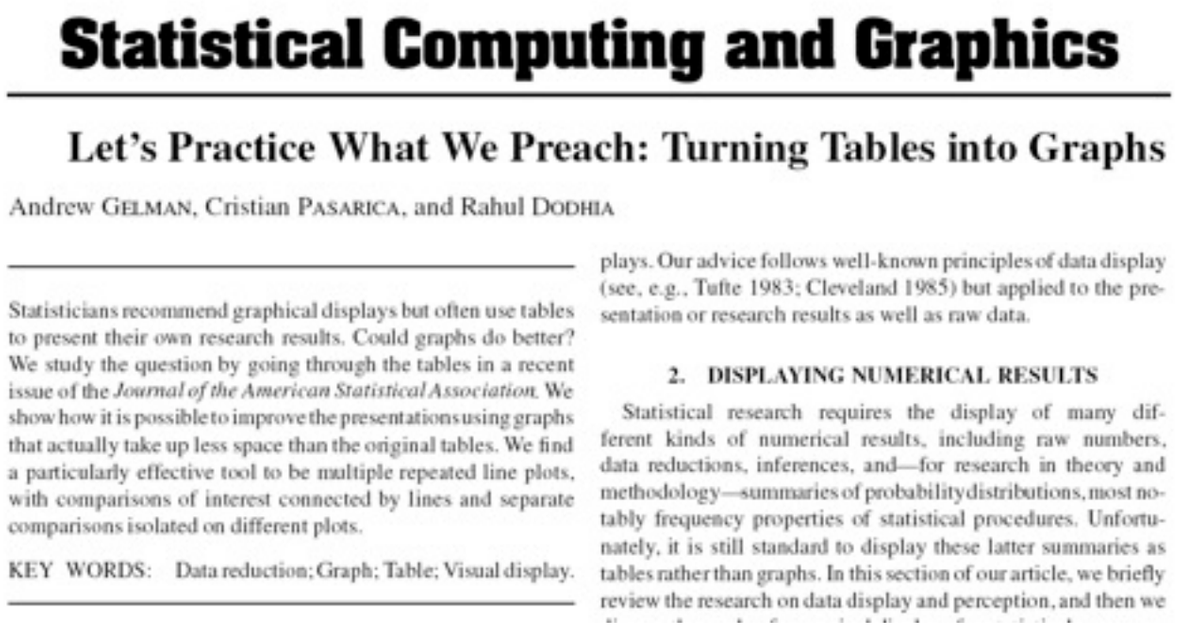
With this figure, it easy to recover which pairs are most different and what the actual p-value is (approximately).

There's always room for improvement

- Better size for density plots ... make levelplot rectangular not square? How to make sizing more compatible?
- Mapping of p-values to colors, i.e. can I improve on the probit?
- Color ramp ... can I, should I make it easier to read off stat. sig. pairs at a <alpha> level?
- Put the density plot in those wasted blank diagonal squares ... would be awesome but a pain to do?

My goals in this and all plots

- Don't chicken out on "scaling up".
- Try to replace a table of numbers with a picture.
- Facilitate comparisons
- Facilitate the identification of trends
- Recommended reference: Gelman A, Pasarica C, Dodhia R. "Let's Practice What We Preach: Turning Tables into Graphs". The American Statistician, Volume 56, Number 2, 1 May 2002 , pp. 121-130(10). via JSTOR



Statistical Computing and Graphics

Let's Practice What We Preach: Turning Tables into Graphs

Andrew GELMAN, Cristian PASARICA, and Rahul DODHIA

Statisticians recommend graphical displays but often use tables to present their own research results. Could graphs do better? We study the question by going through the tables in a recent issue of the *Journal of the American Statistical Association*. We show how it is possible to improve the presentations using graphs that actually take up less space than the original tables. We find a particularly effective tool to be multiple repeated line plots, with comparisons of interest connected by lines and separate comparisons isolated on different plots.

KEY WORDS: Data reduction; Graph; Table; Visual display.

plays. Our advice follows well-known principles of data display (see, e.g., Tufte 1983; Cleveland 1985) but applied to the presentation or research results as well as raw data.

2. DISPLAYING NUMERICAL RESULTS

Statistical research requires the display of many different kinds of numerical results, including raw numbers, data reductions, inferences, and—for research in theory and methodology—summaries of probability distributions, most notably frequency properties of statistical procedures. Unfortunately, it is still standard to display these latter summaries as tables rather than graphs. In this section of our article, we briefly review the research on data display and perception, and then we

2.2 The Goals and Principles of Statistical Data Display

Constructing the graphs shown in this article actually took a lot of time and several iterations, with each step provoking further thought on the motivation behind various techniques of graphical display. Here, we attempt to identify some of the key ideas, with the hope of making the task easier for future researchers.

Tables are more effective if the goal is to read off exact numbers. However, the interest in a statistical paper typically lies in comparisons, not absolute numbers. For example, there is no reason for the reader to care that the standard error for a particular parameter estimate is 0.029 as computed under a certain method and 0.054 under another method. Rather, the point is that the second number is almost twice as high. When dozens of such potential comparisons are possible, they can be seen much more clearly in a well-chosen graphical display than in a table (see Figure 3). The idea of comparisons provides a theoretical framework for statistical graphics (as is implicit in Tukey 1972, 1977).

from Gelman et al (2002)

Raters' characterization of sentences (absolute score range)	*Percent*
Negative (1–1.9)	23.9
Neutral (2–2.9)	52.2
Positive ($\geq$ 3)	23.9
Total	100.0

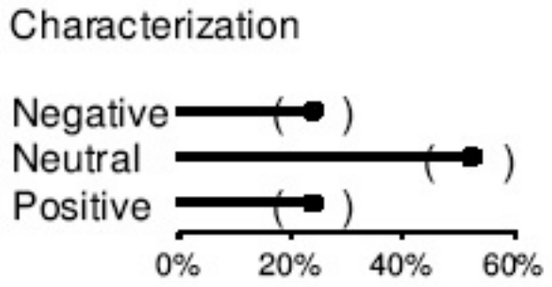
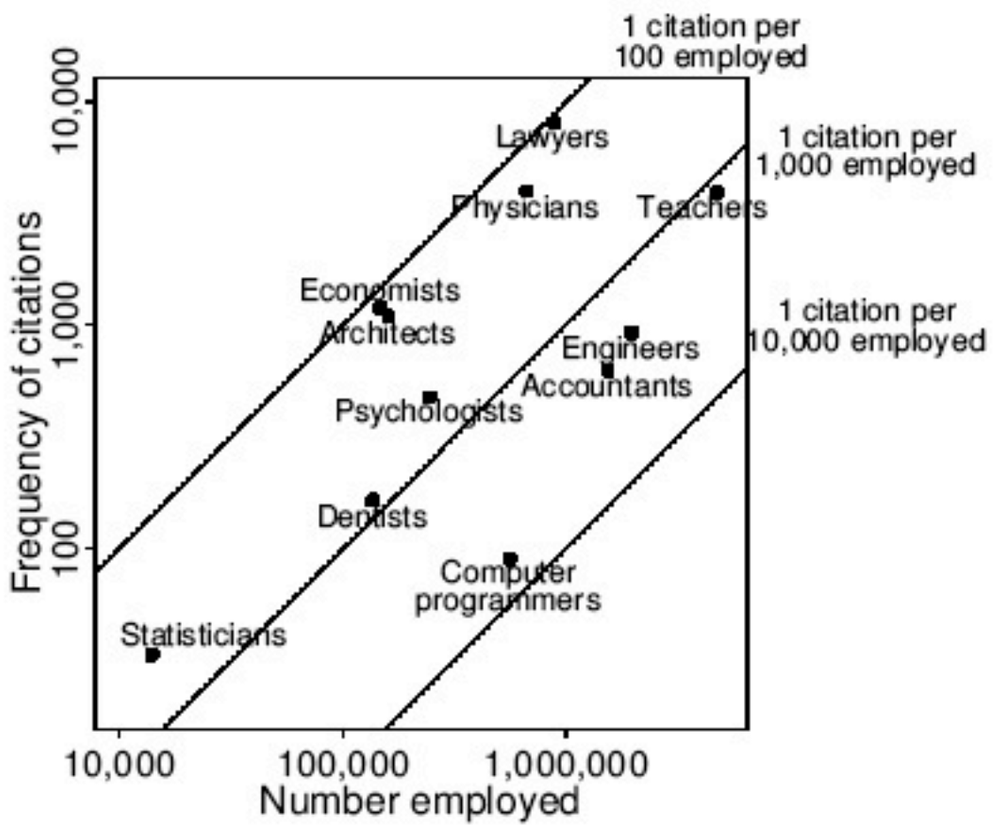


Figure 1. Top panel: Table from Ellenberg (2000) shows the relative frequencies of three categories in a set of 46 ratings of sentences. Bottom panel: Graphical display allows direct comparison without distractions of irrelevant decimal places. Parentheses show ±1 standard error bounds based on the implicit binomial distribution with n = 46.

Profession	*Frequency of recent citations*	*1996 total employed (1,000)*	*Relative frequency*
Lawyers	8101	880	9.2
Economists	1201	148	8.1
Architects	1097	160	6.9
Physicians	3989	667	6.0
Statisticians	34	14	2.4
Psychologists	479	245	2.0
Dentists	165	137	1.2
Teachers (not university)	3938	4724	0.8
Engineers	934	1960	0.5
Accountants	628	1538	0.4
Computer programmers	91	561	0.2
Total	20,657	11,034	1.9



from Gelman et al (2002)

A common first reaction to rants about data display is: Yeah, yeah, I know that already. But, as we have seen, we—the statistical profession—don't know it yet! If the world's leading statistical journal doesn't do it right, there is obviously still room for progress. And the first steps are recognizing why the problem exists and forming a step-by-step plan to fix it. The steps we recommend, for most data displays in statistical research, are: (1) identify the key comparisons of interest; (2) display these on small individual plots with comparison lines or axes where appropriate; and (3) establish enough control over the graphical display so that small legible plots can be juxtaposed as necessary.

The graphs in this article are extremely simple, and serious exploratory data analysis benefits from many elaborations, including color, multiple linked (trellis) plots, and dynamic displays (see, for a start, Chambers et al. 1983; Cleveland 1985, 1993). We

from Gelman et al (2002)

Something to think about in your project and future work ... can you complement or replace a big table of numbers with a figure? Can you employ multiple methodologies and assess how same/different the results are?

Now ... moving beyond the “classical” tests and introducing the bootstrap.

CLASS ENDED HERE.