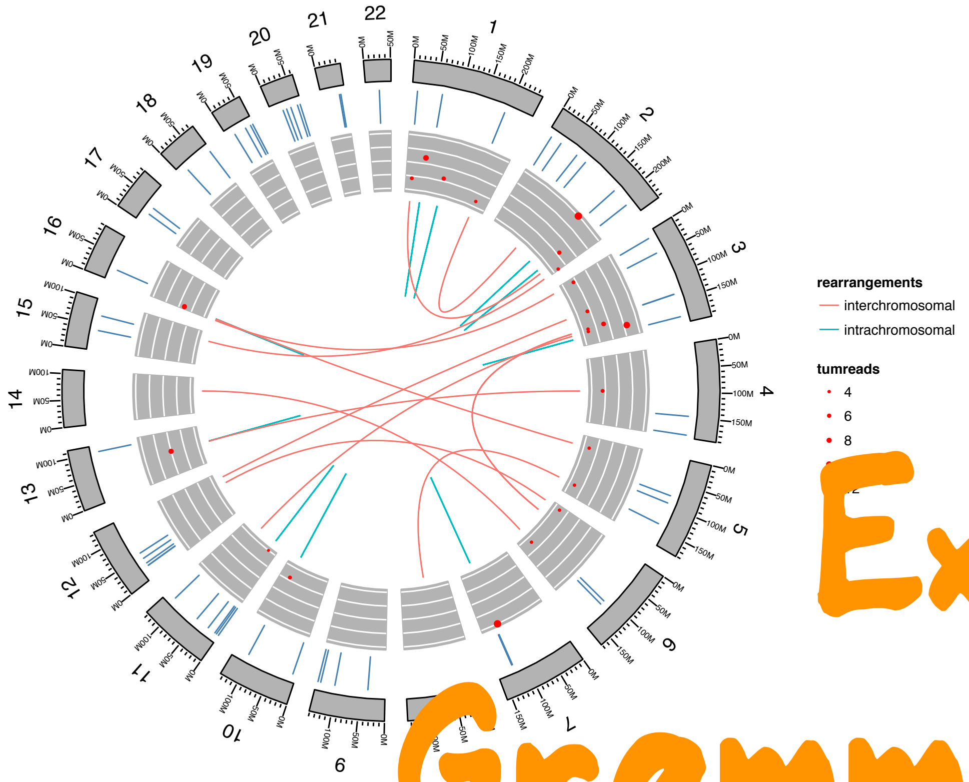
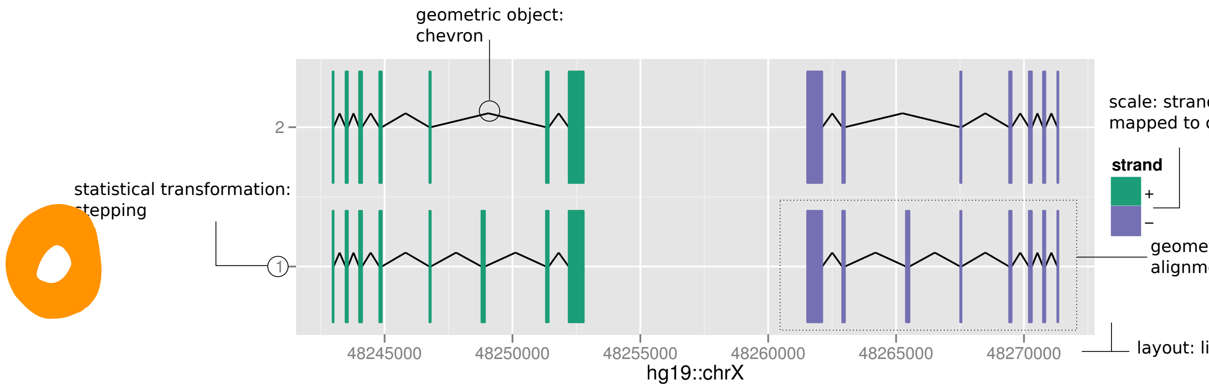




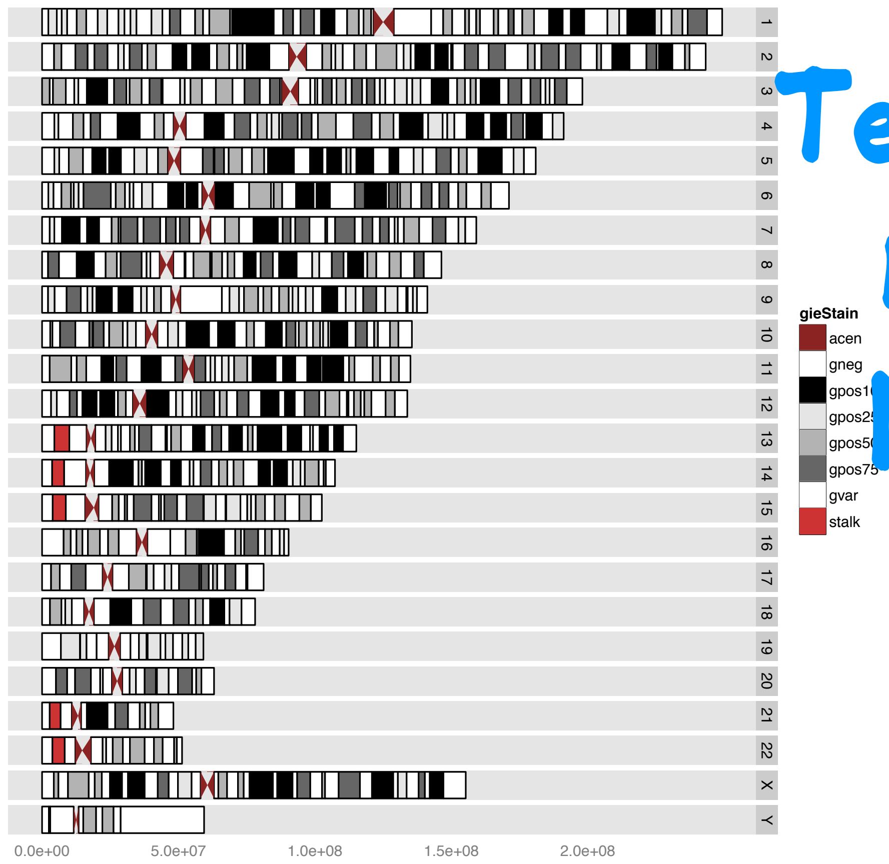
ggbio
Extending the Grammar of Graphics to Genomic Data

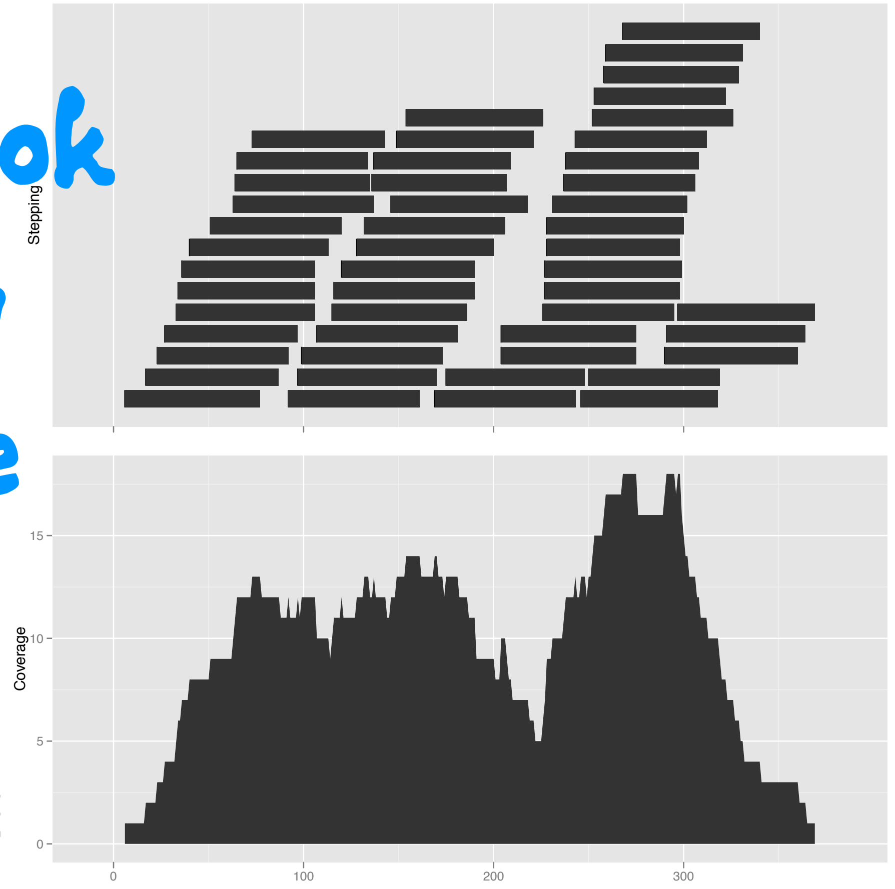
Tengfei Yin, Di Cook

Iowa State University

Michael Lawrence

Genentech



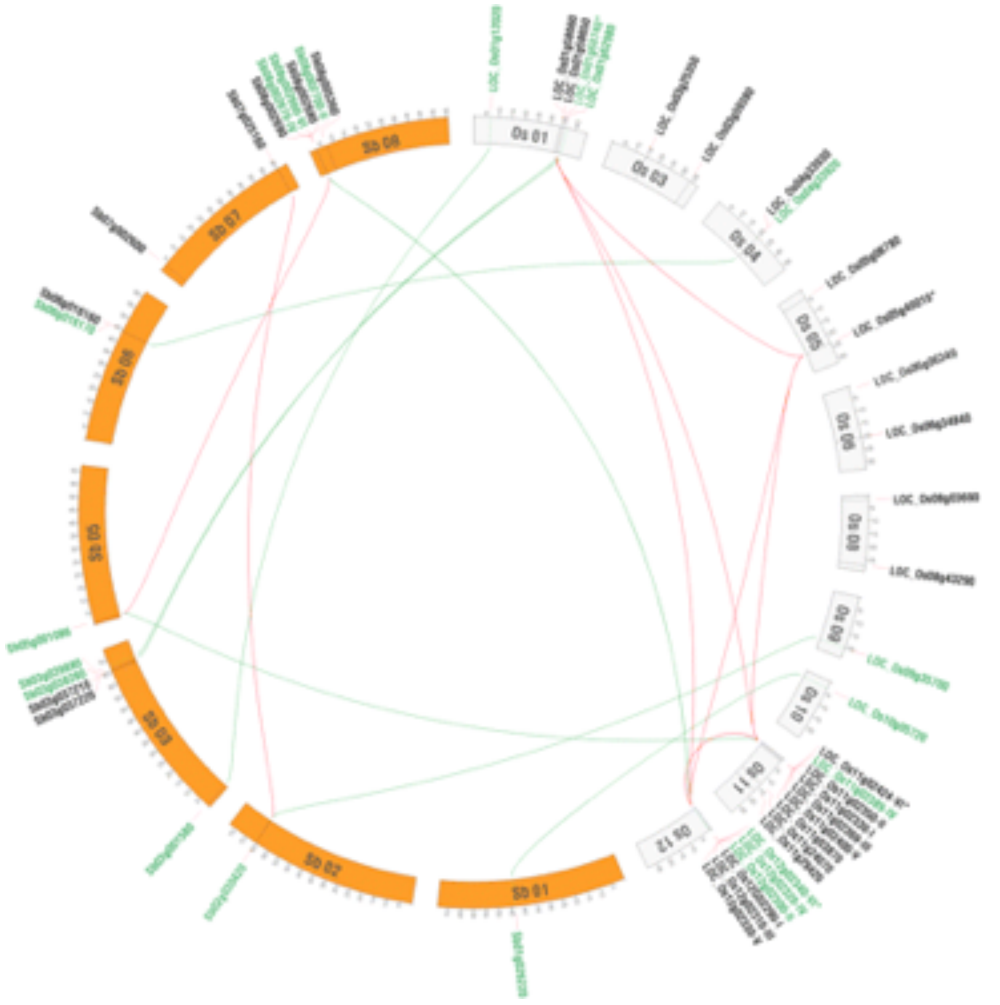


terface 2012, Rice Universit

Motivation

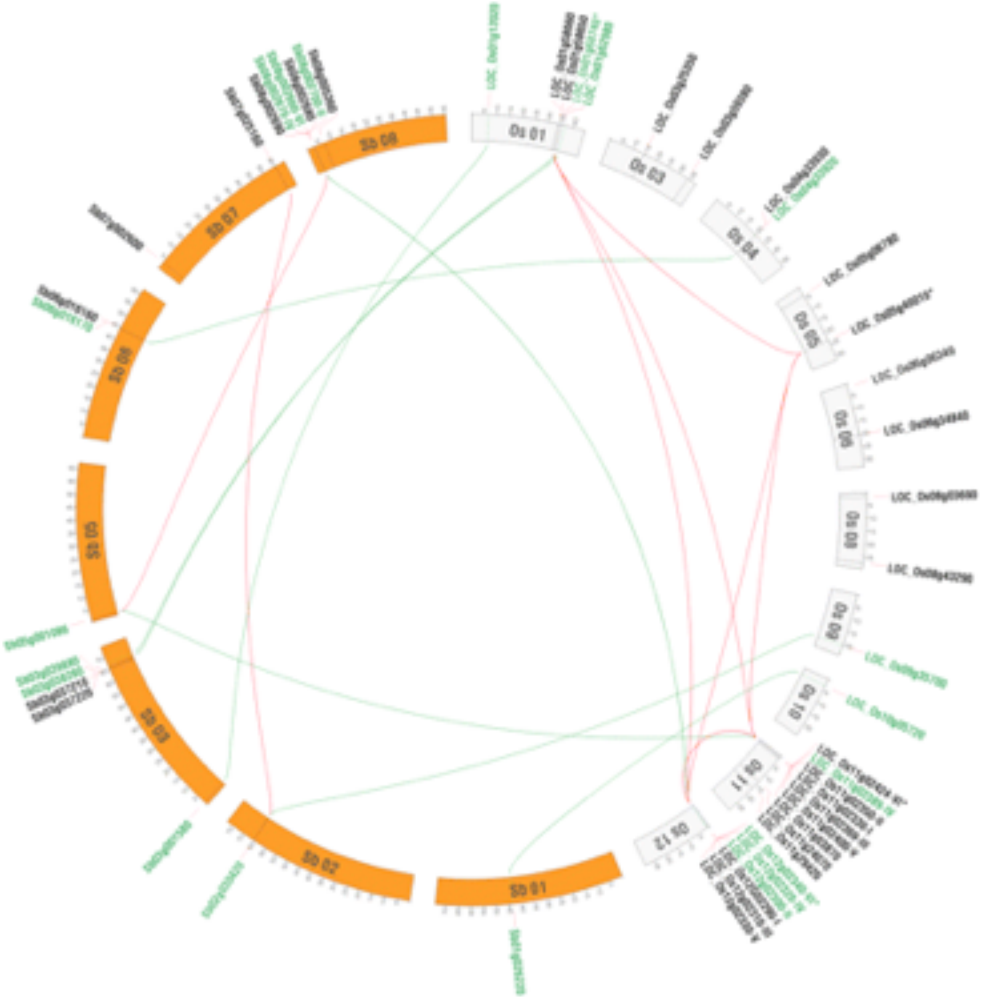
- Lots of tools exist for displaying genomic data
- Many different packages, many standalone, many different data standards

Motivation


Os 01
Os 03
Os 04
Os 05
Os 06
Os 08
Os 09
Os 10
Os 11
Os 12
Sb 01
Sb 02
Sb 03
Sb 05
Sb 06
Sb 07
Sb 08

Motivation

Circos

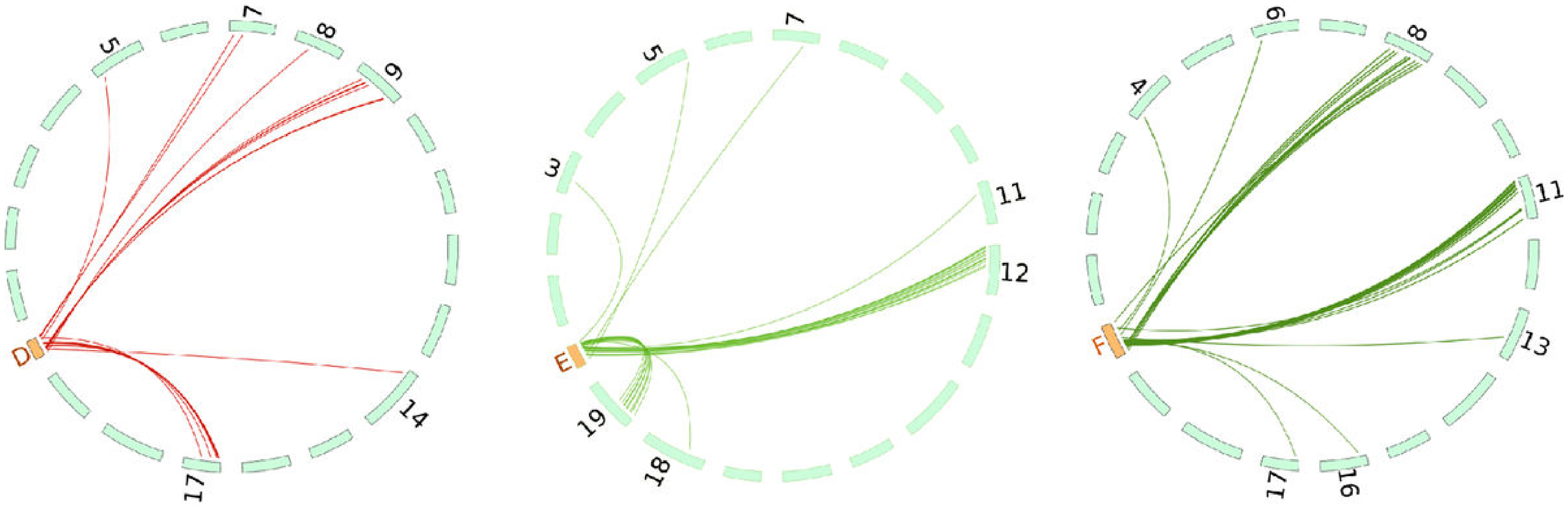


Motivation

Circos

Motivation

Circos

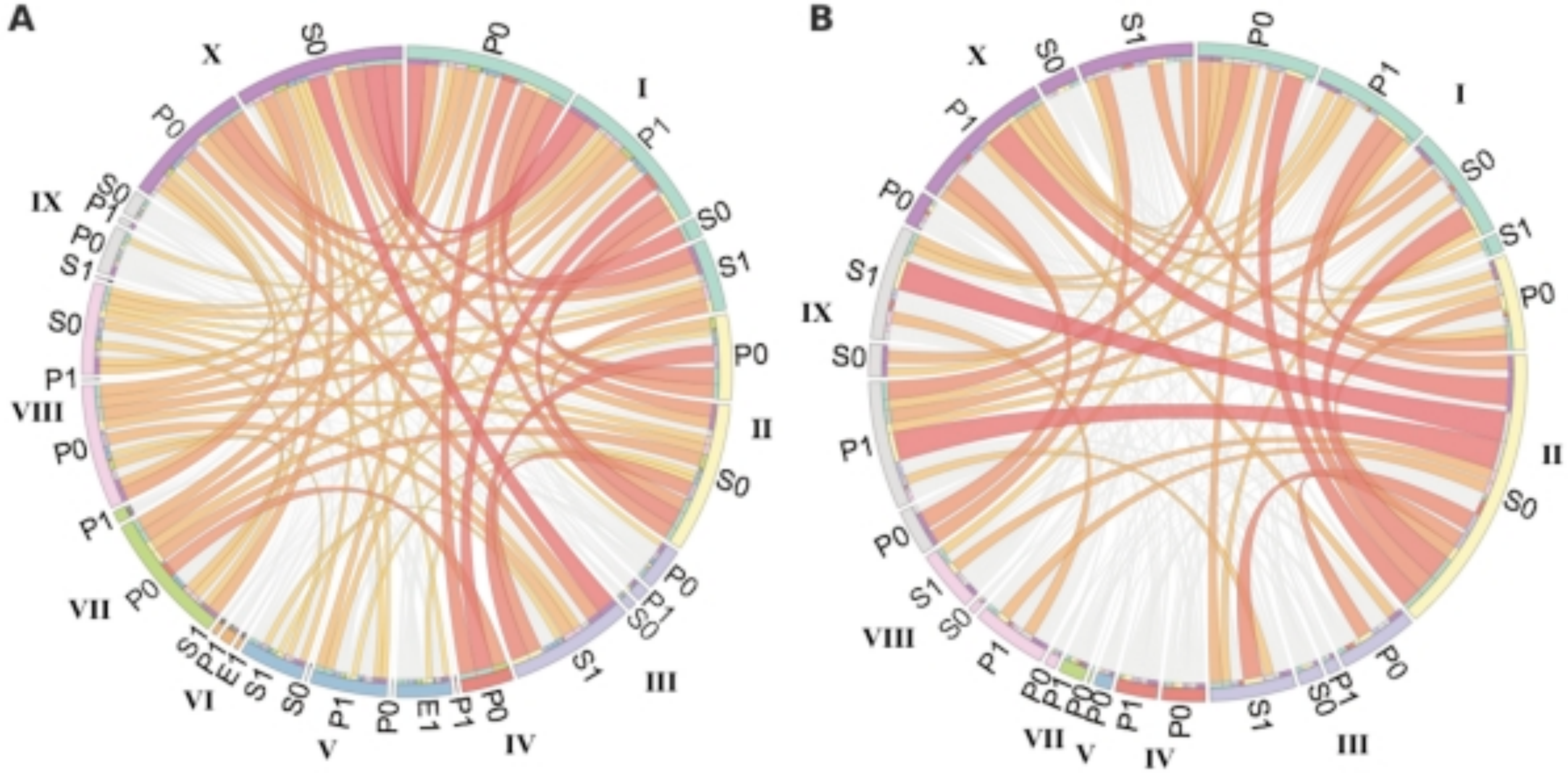


Motivation

Circos

Motivation

Circos



Motivation

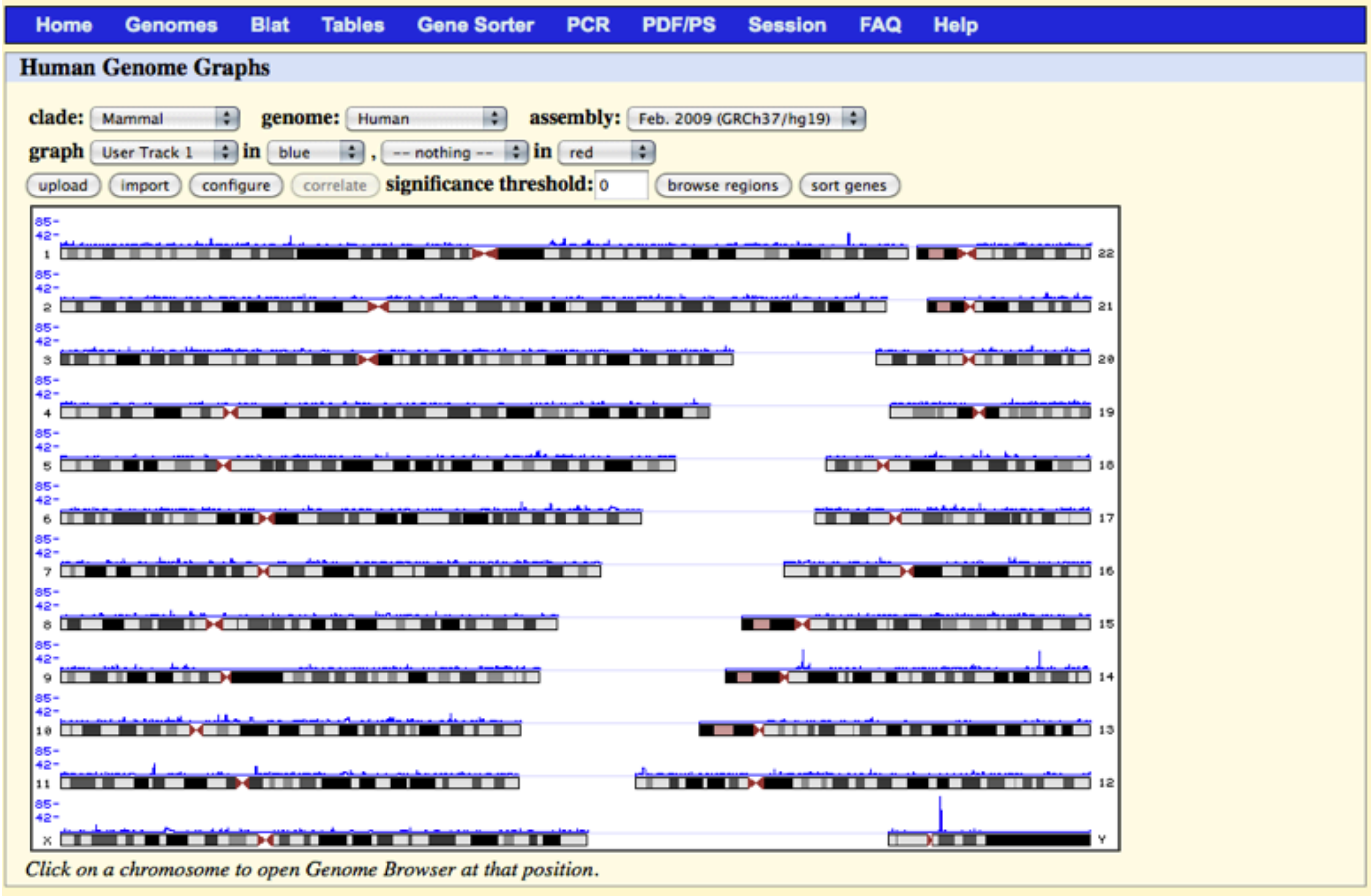
Circos

Motivation

Circos

- Need construct a central and many other configuration files from scratch, learning curve is very high
- Adding legend not easy
- Cannot map aesthetics to certain variables

Motivation



Human Genome Graphs

clade: Mammal genome: Human assembly: Feb. 2009 (GRCh37/hg19)

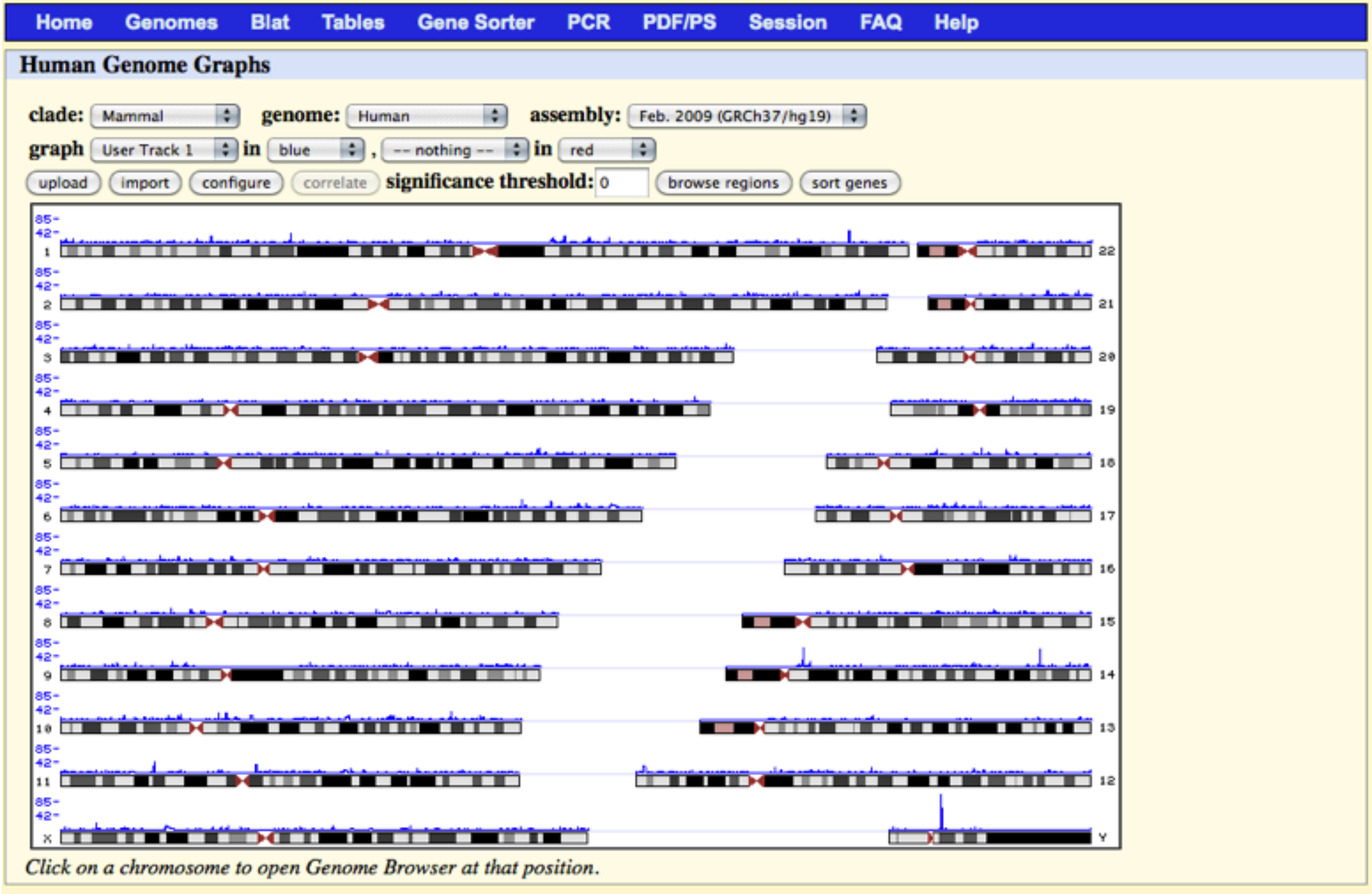
graph User Track 1 in blue , -- nothing -- in red

significance threshold: 0

Click on a chromosome to open Genome Browser at that position.

Motivation

UCSC Genome Browser



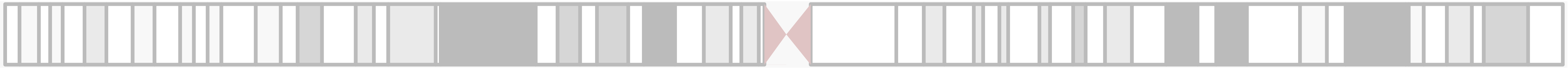
Human Genome Graphs

clade: Mammal genome: Human assembly: Feb. 2009 (GRCh37/hg19)

graph User Track 1 in blue , -- nothing -- in red

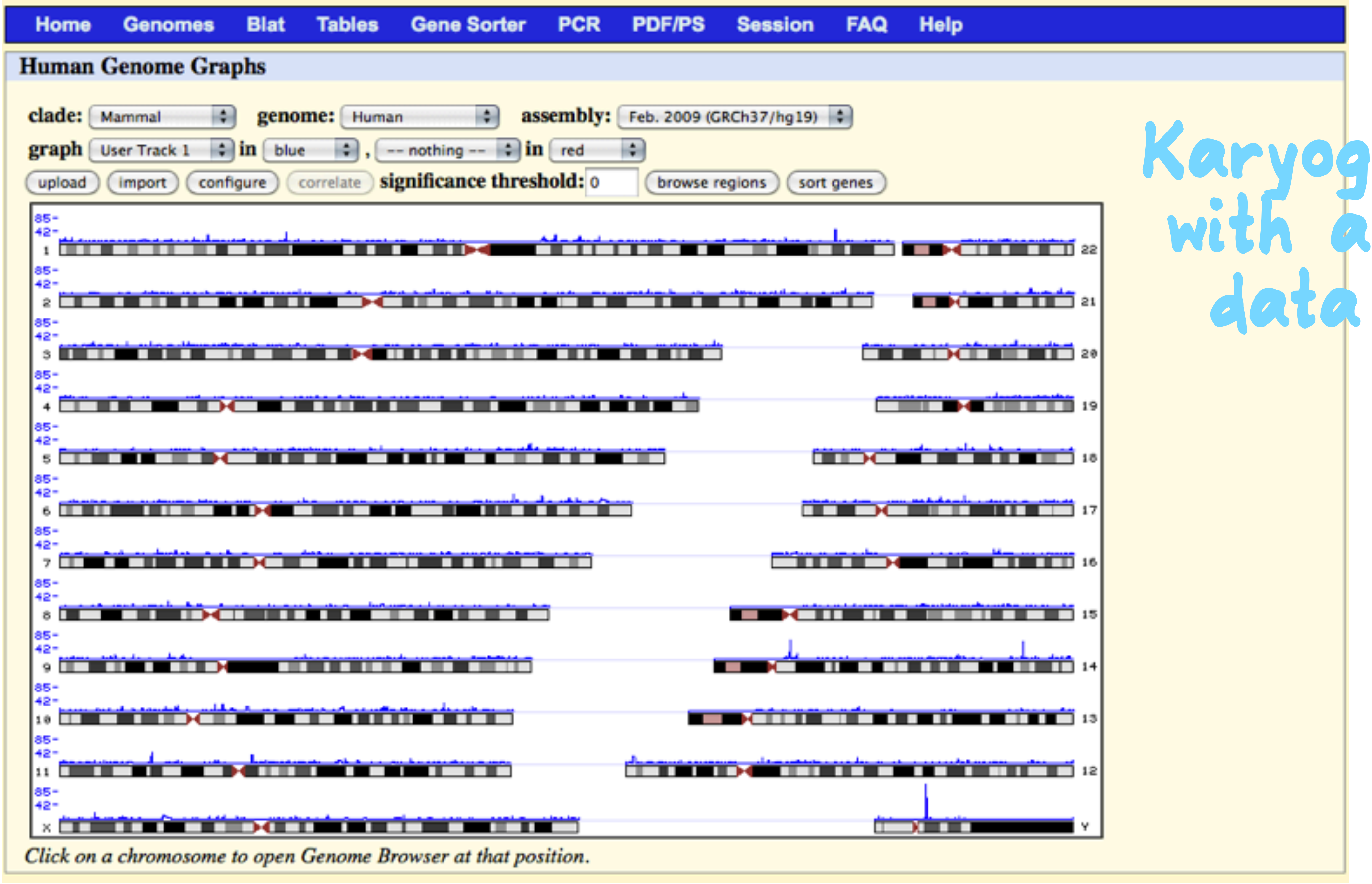
significance threshold: 0

Click on a chromosome to open Genome Browser at that position.



Motivation

UCSC Genome Browser



Human Genome Graphs

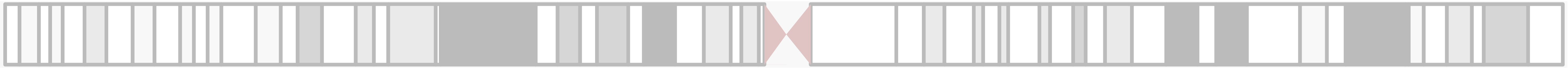
clade: Mammal genome: Human assembly: Feb. 2009 (GRCh37/hg19)

graph User Track 1 in blue , -- nothing -- in red

significance threshold: 0

Click on a chromosome to open Genome Browser at that position.

Karyogram view, with associated data plotted



Motivation

UCSC Genome Browser

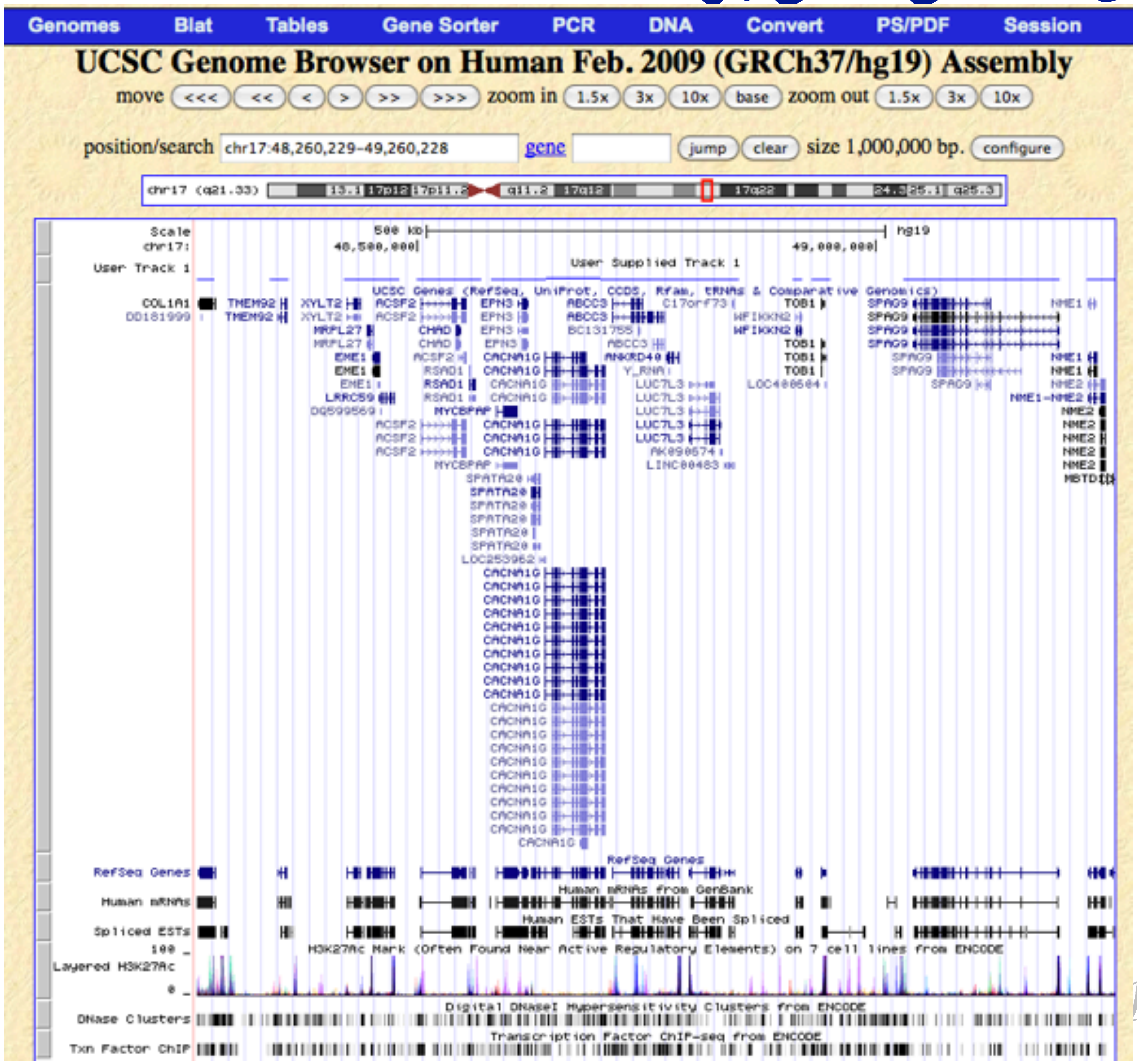
Karyogram view,
with associated
data plotted

Motivation

UCSC Genome Browser

Motivation

UCSC Genome Browser

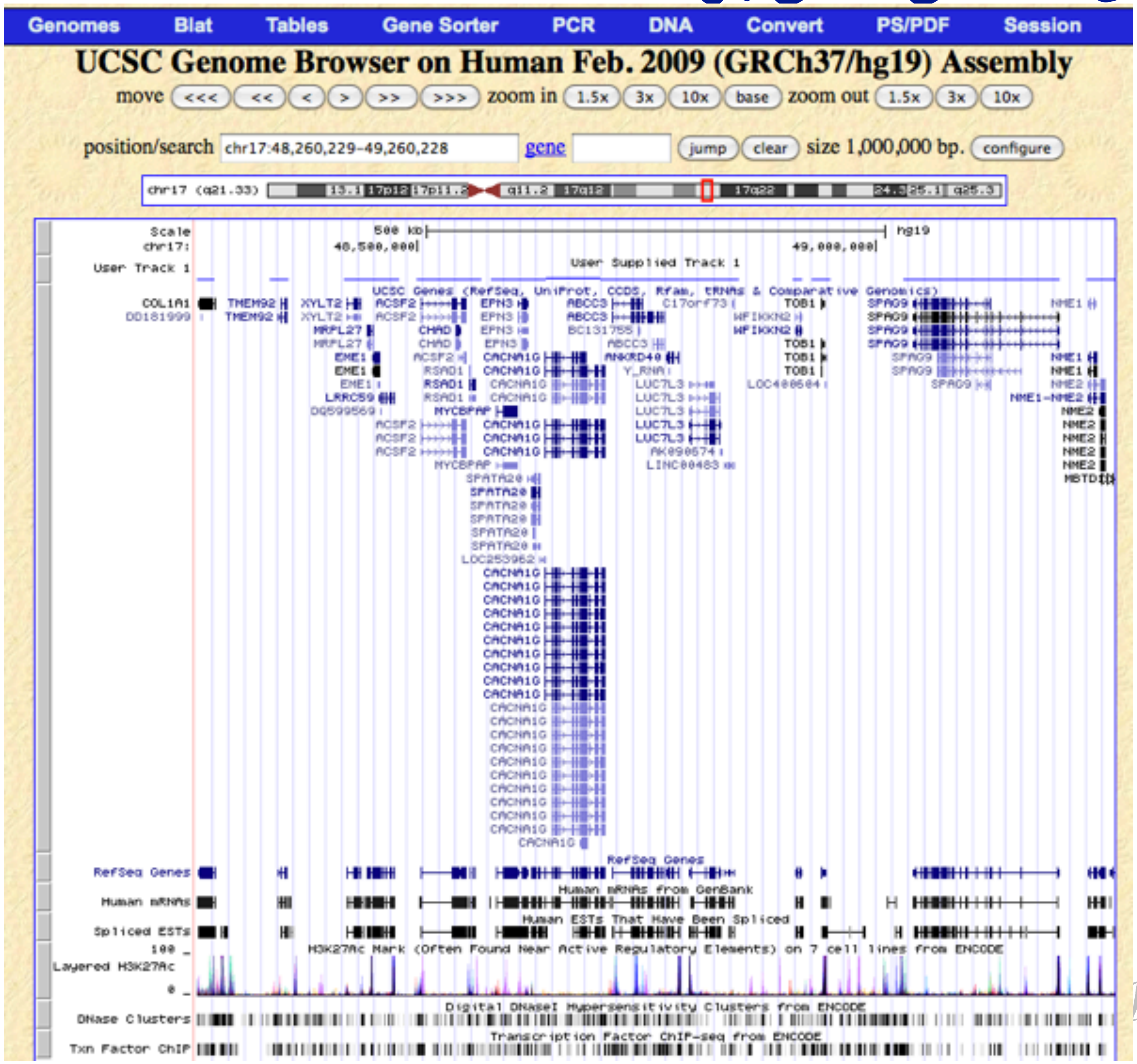


UCSC Genome Browser on Human Feb. 2009 (GRCh37/hg19) Assembly

position/search chr17:48,260,229-49,260,228 size 1,000,000 bp.

Motivation

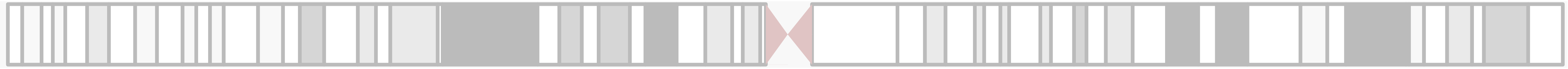
UCSC Genome Browser



UCSC Genome Browser on Human Feb. 2009 (GRCh37/hg19) Assembly

position/search chr17:48,260,229-49,260,228 size 1,000,000 bp.

Logical zoom, all we know about this genetic code



Motivation

UCSC Genome Browser

Logical zoom, all we know about this genetic code

Motivation

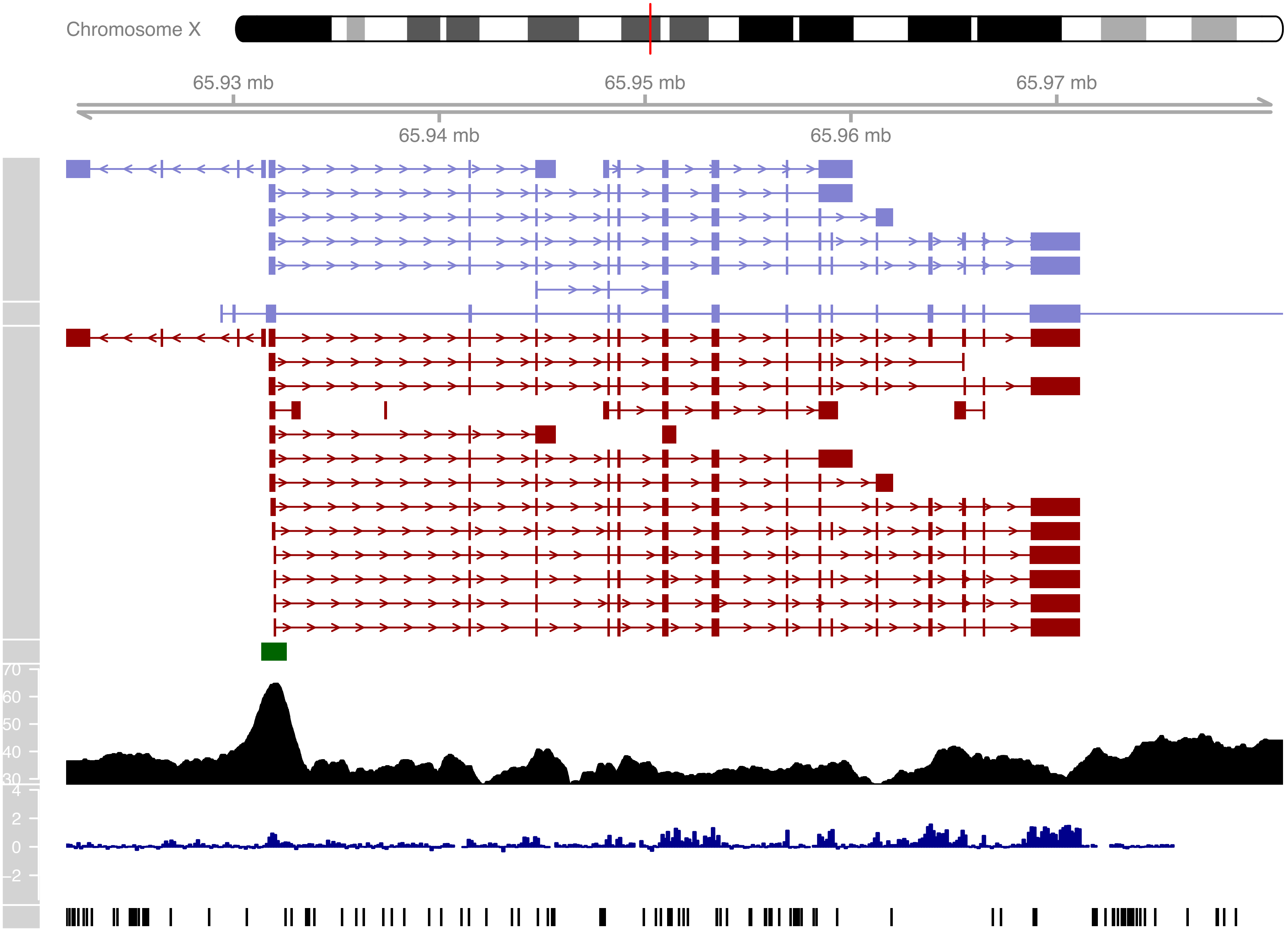
UCSC Genome Browser

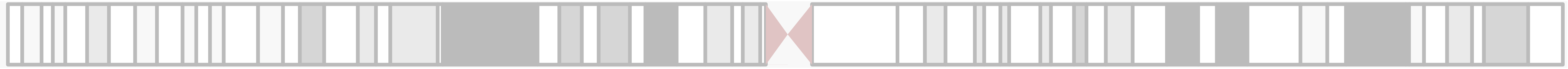
Motivation

UCSC Genome Browser

- Very commonly used, very popular
- Gives broadly applicable, generic, but narrow selection of plot choices
- No operations on genomic ranges views to facilitate perception of structure

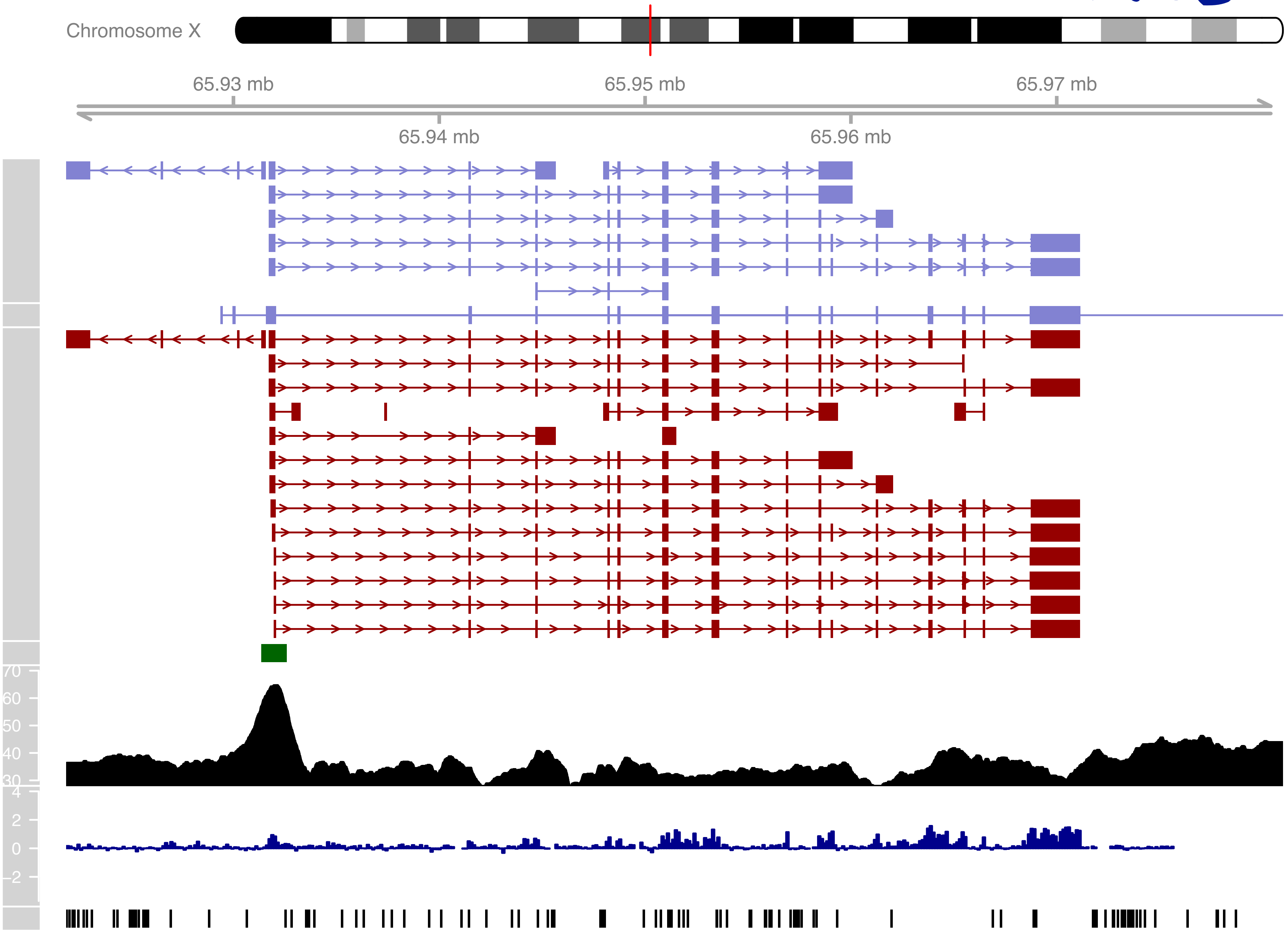
Motivation


Chromosome X
65.93 mb
65.95 mb
65.97 mb
65.94 mb
65.96 mb
70
60
50
40
30
4
2
0
−2



Motivation

Gviz (Hahne et al)



Motivation

Gviz (Hahne et al)

Motivation

Gviz (Hahne et al)

- Pretty good!
- Incorporated with R, and R data structures
- Uses grid (low level) graphics, very flexible, but not leveraging tools like ggplot2

Outline

- What is the grammar of graphics?
- How it is extended for genomic data.
- Examples
- Next steps: interactive graphics

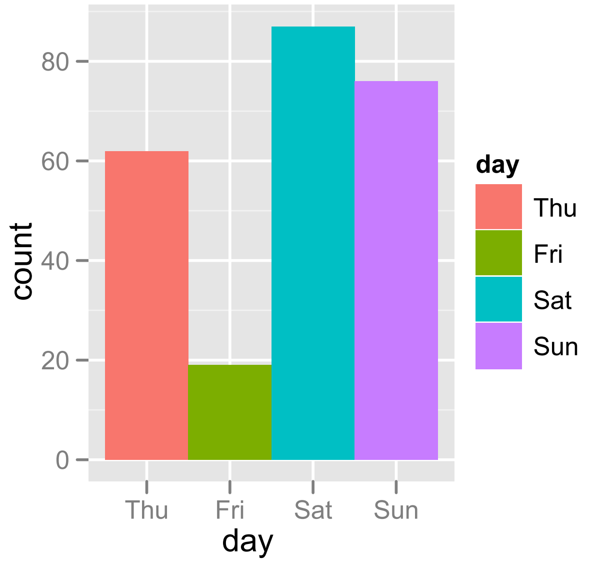
Grammar

- Grammar forms the foundation of a language. It is a set of structural rules that govern composition.
- For graphics, it provides a way to construct a plot in a common form, and enables clarification of similarities and differences between plots.

Grammar (ggplot2)

Bar chart

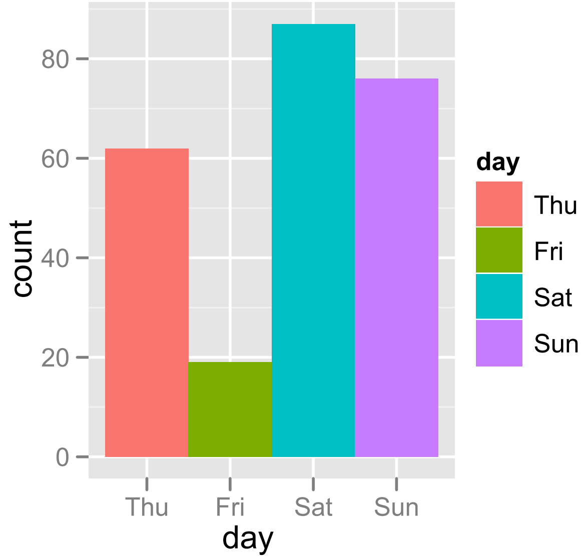
```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```



Grammar (ggplot2)

Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```



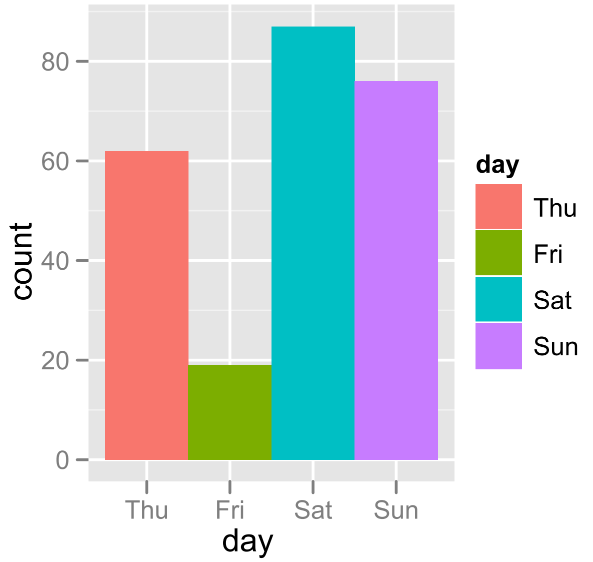
Pie chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1) +
   coord_polar()
```

Grammar (ggplot2)

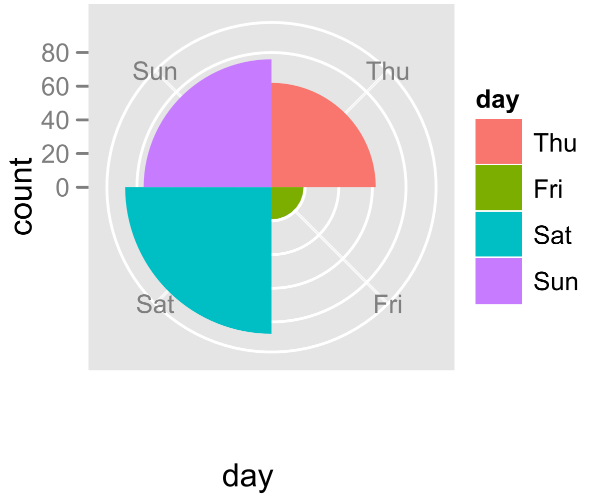
Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```



Pie chart

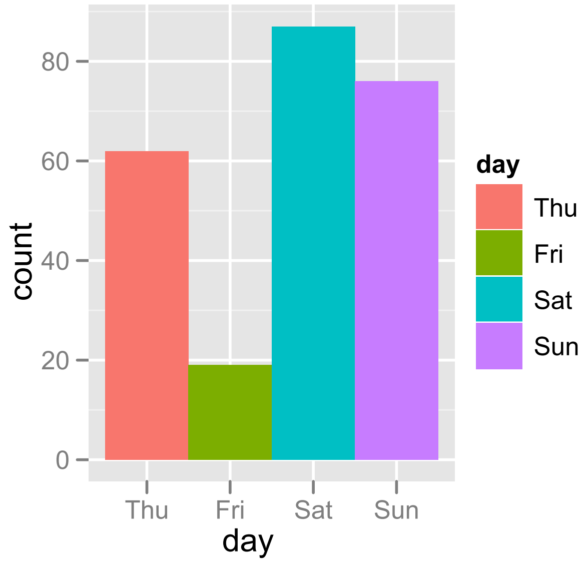
```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1) +
   coord_polar()
```



Grammar (ggplot2)

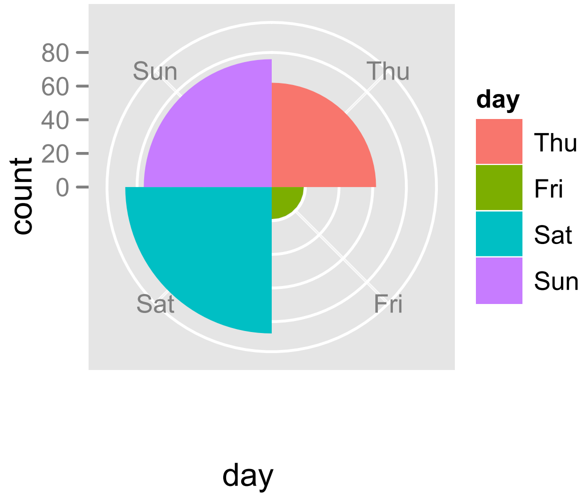
Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```



~~Pie chart~~

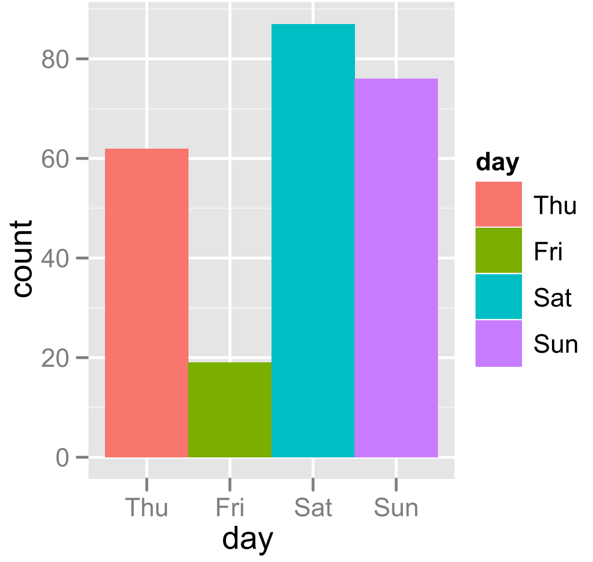
```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1) +
   coord_polar()
```



Grammar (ggplot2)

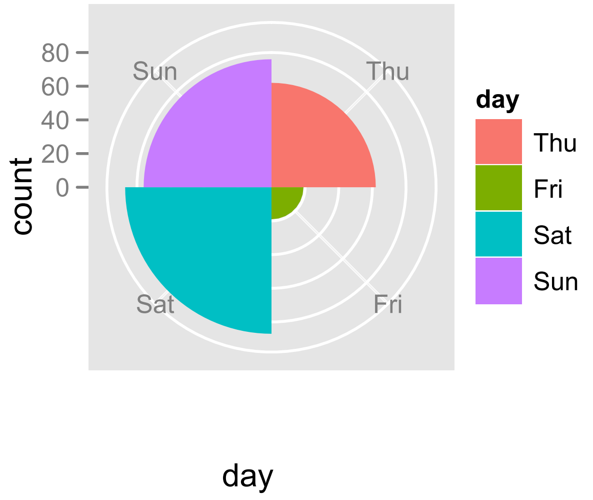
Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```



Pie chart

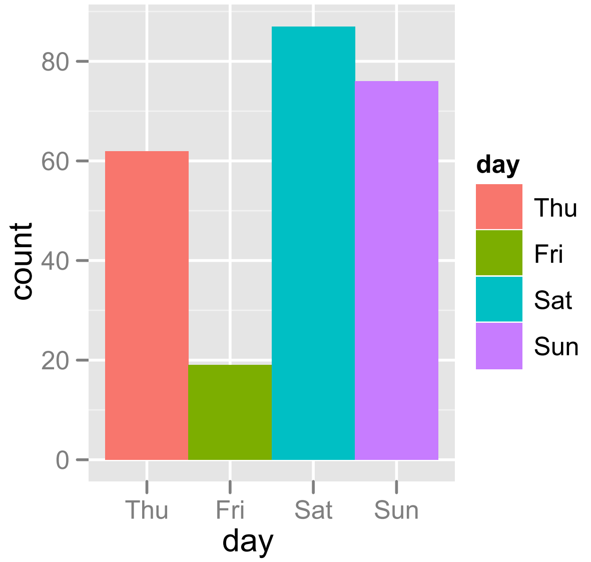
```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1) +
   coord_polar()
```

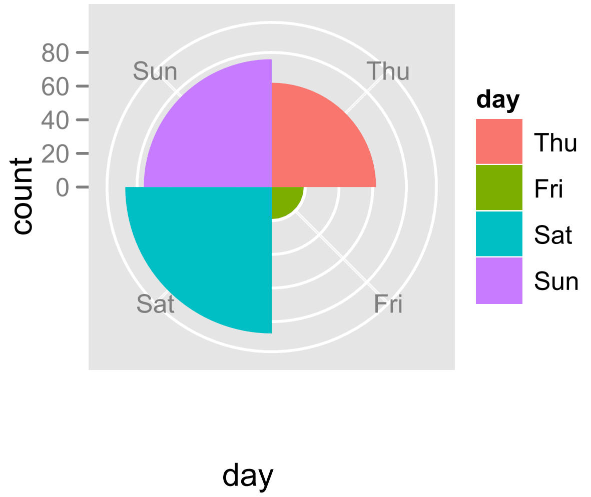


Grammar (ggplot2)

Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```

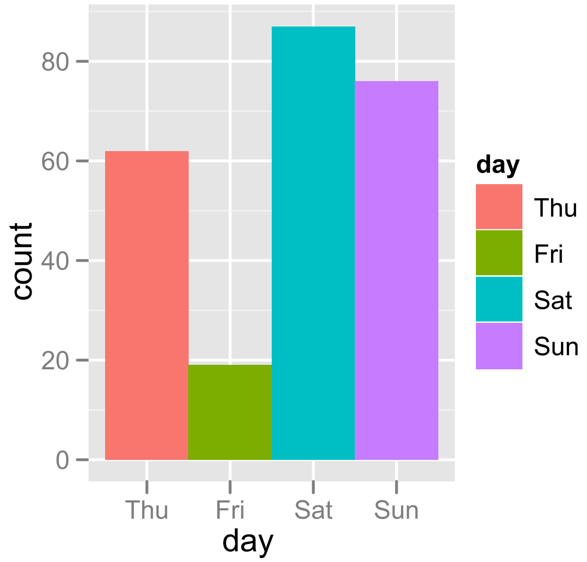




Grammar (ggplot2)

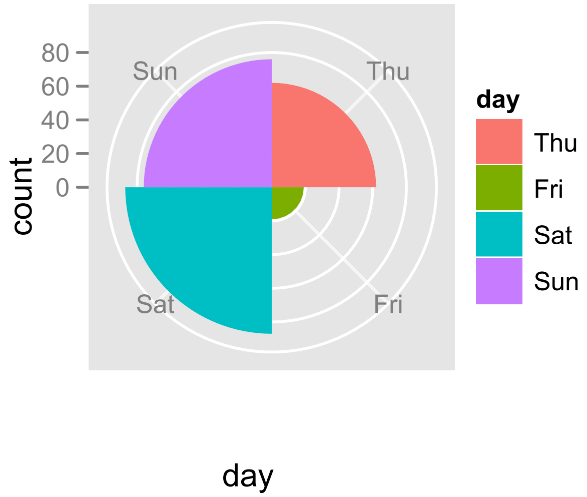
Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```



Rose plot/Coxcomb

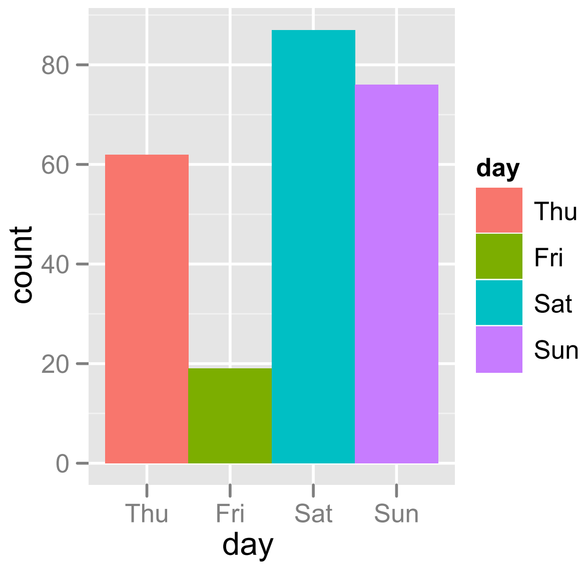
```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1) +
   coord_polar()
```

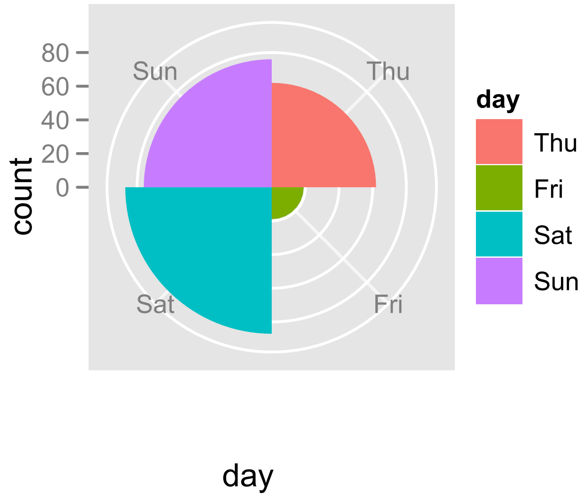


Grammar (ggplot2)

Bar chart

```
ggplot(data=tips,
   aes(x=day, fill=day)) +
   geom_bar(width=1)
```

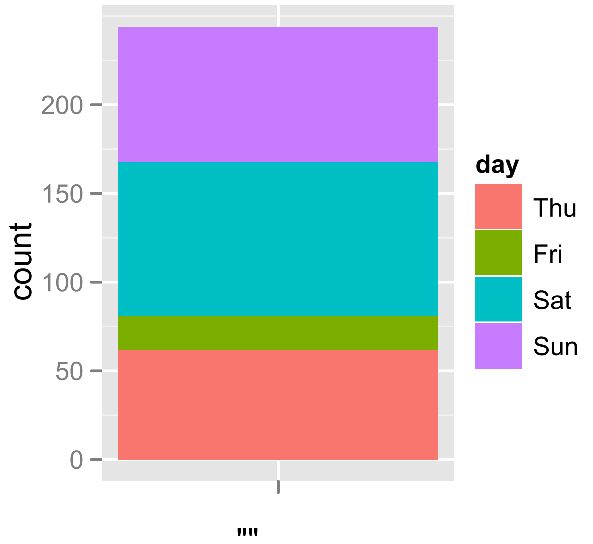




Grammar (ggplot2)

Stacked bar chart

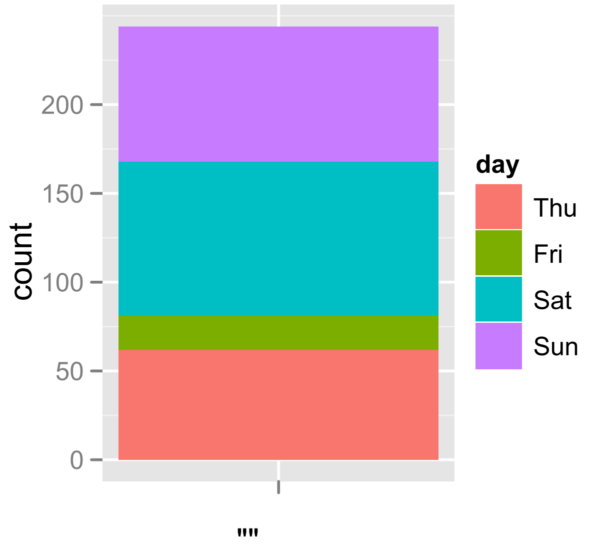
```
ggplot(data=tips,
   aes(x=””, fill=day)) +
   geom_bar(width=1)
```



Grammar (ggplot2)

Stacked bar chart

```
ggplot(data=tips,
   aes(x=””, fill=day)) +
   geom_bar(width=1)
```



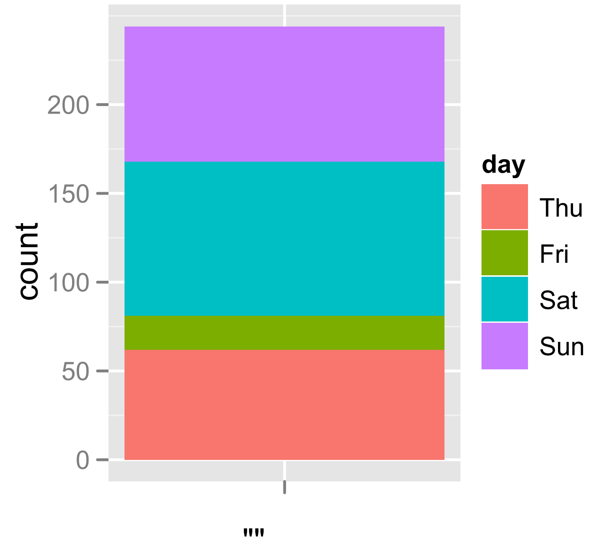
Pie chart

```
ggplot(data=tips,
   aes(x=””, fill=day)) +
   geom_bar(width=1) +
   coord_polar(theta="y")
```

Grammar (ggplot2)

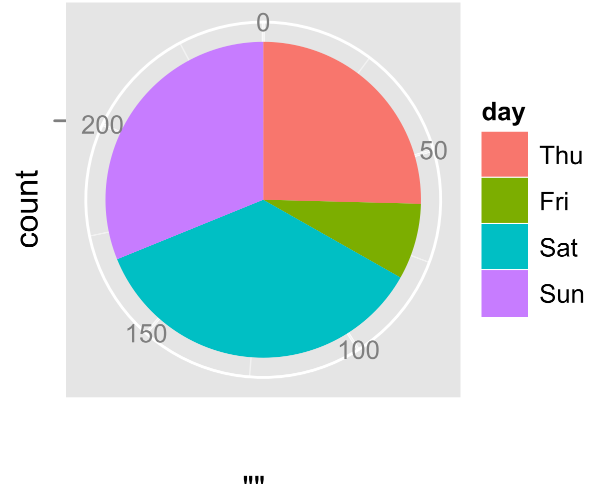
Stacked bar chart

```
ggplot(data=tips,
   aes(x=””, fill=day)) +
   geom_bar(width=1)
```



Pie chart

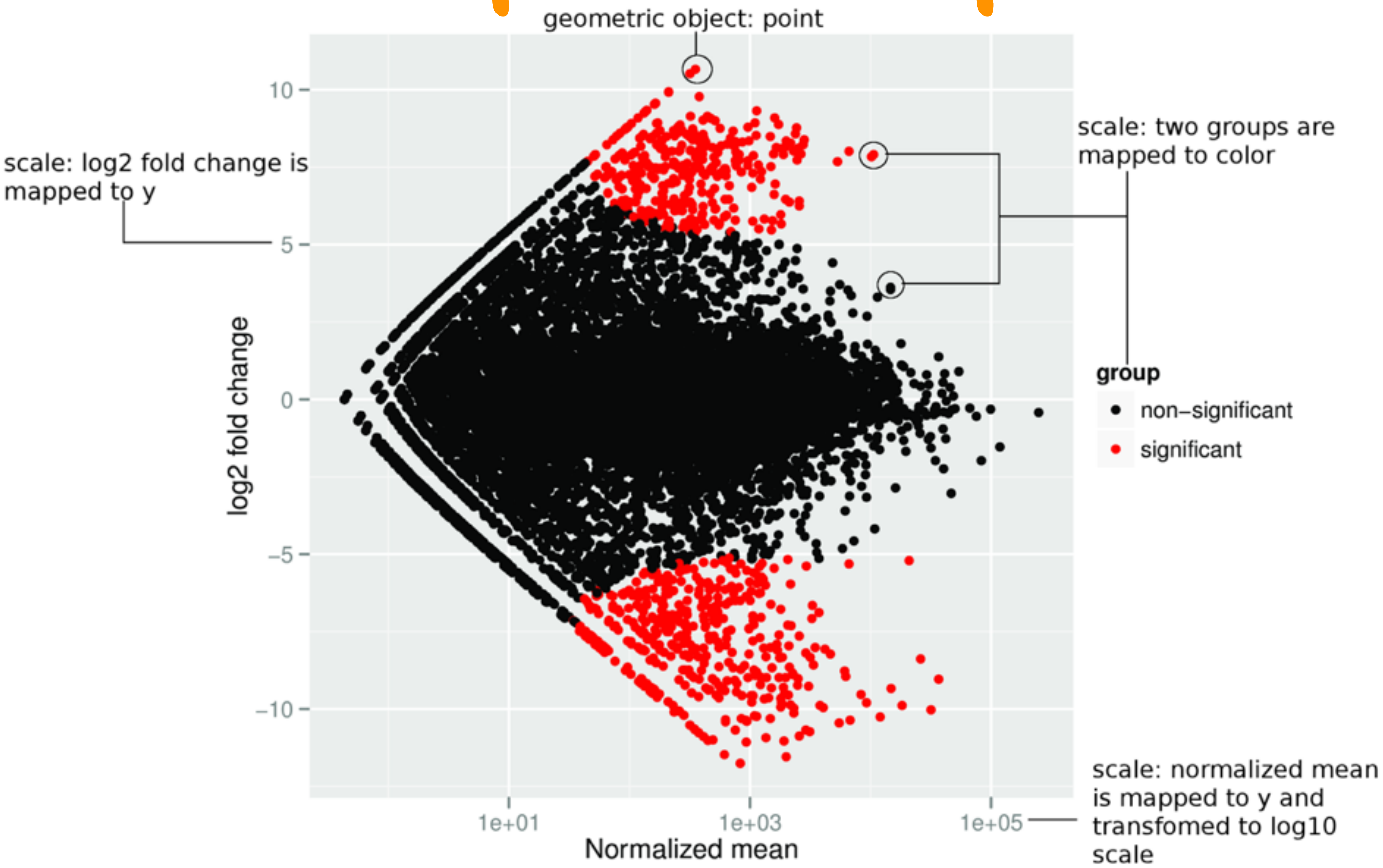
```
ggplot(data=tips,
   aes(x=””, fill=day)) +
   geom_bar(width=1) +
   coord_polar(theta="y")
```



Grammar Elements

- DATA: What is to be plotted
- STAT: Statistical operations to make on data, like binning.
- GEOM: Geometric object, elements to use to displays aspects of the data
- SCALE: Map data to aesthetics to geom
- COORD: Coordinate system to use, eg Cartesian
- (FACET): subset and display

Example: MA plot



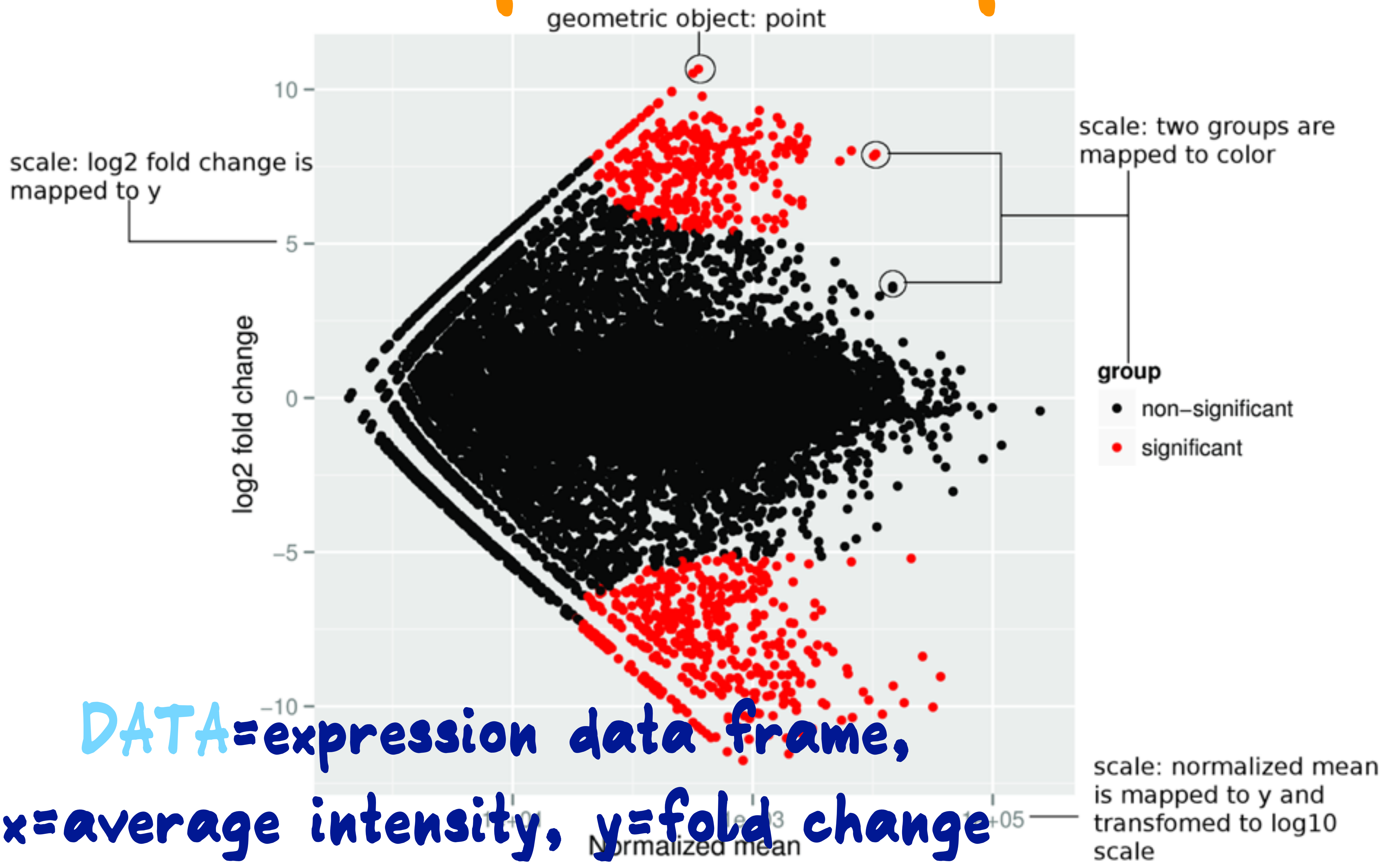
geometric object: point

scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot



geometric object: point

scale: log2 fold change is mapped to y

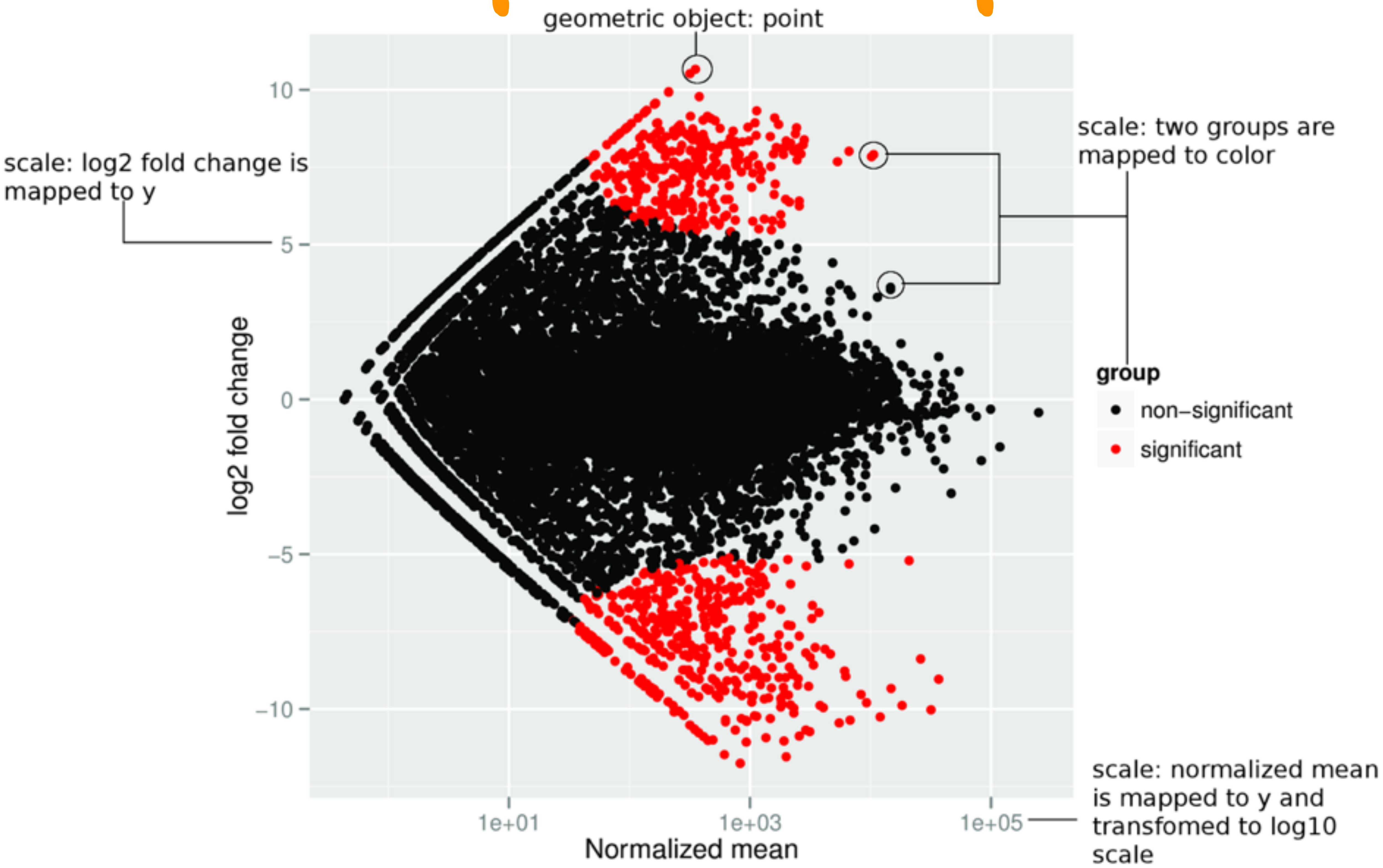
scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

DATA=expression data frame,

x=average intensity, y=fold change

Example: MA plot



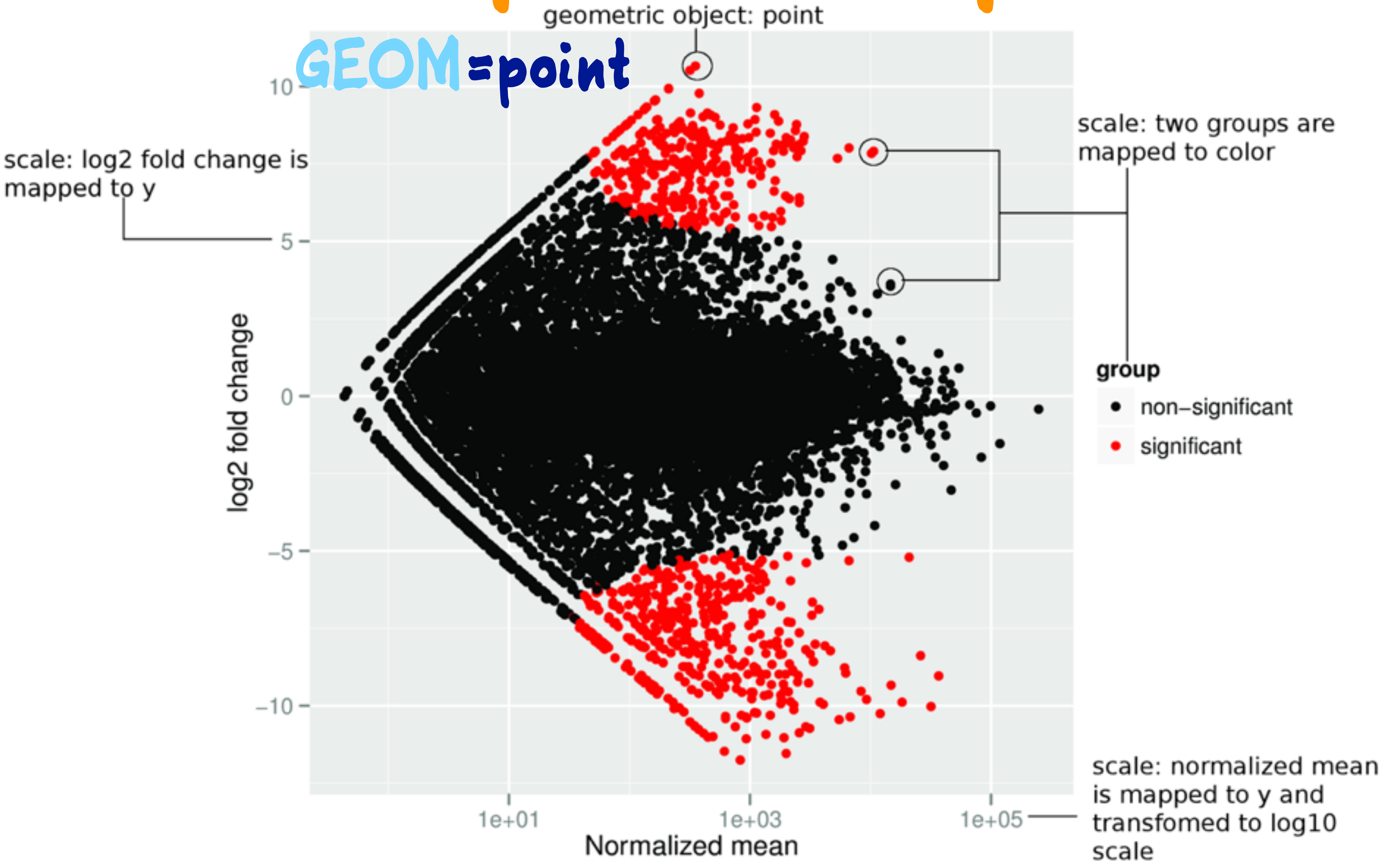
geometric object: point

scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot



GEOM=point

geometric object: point

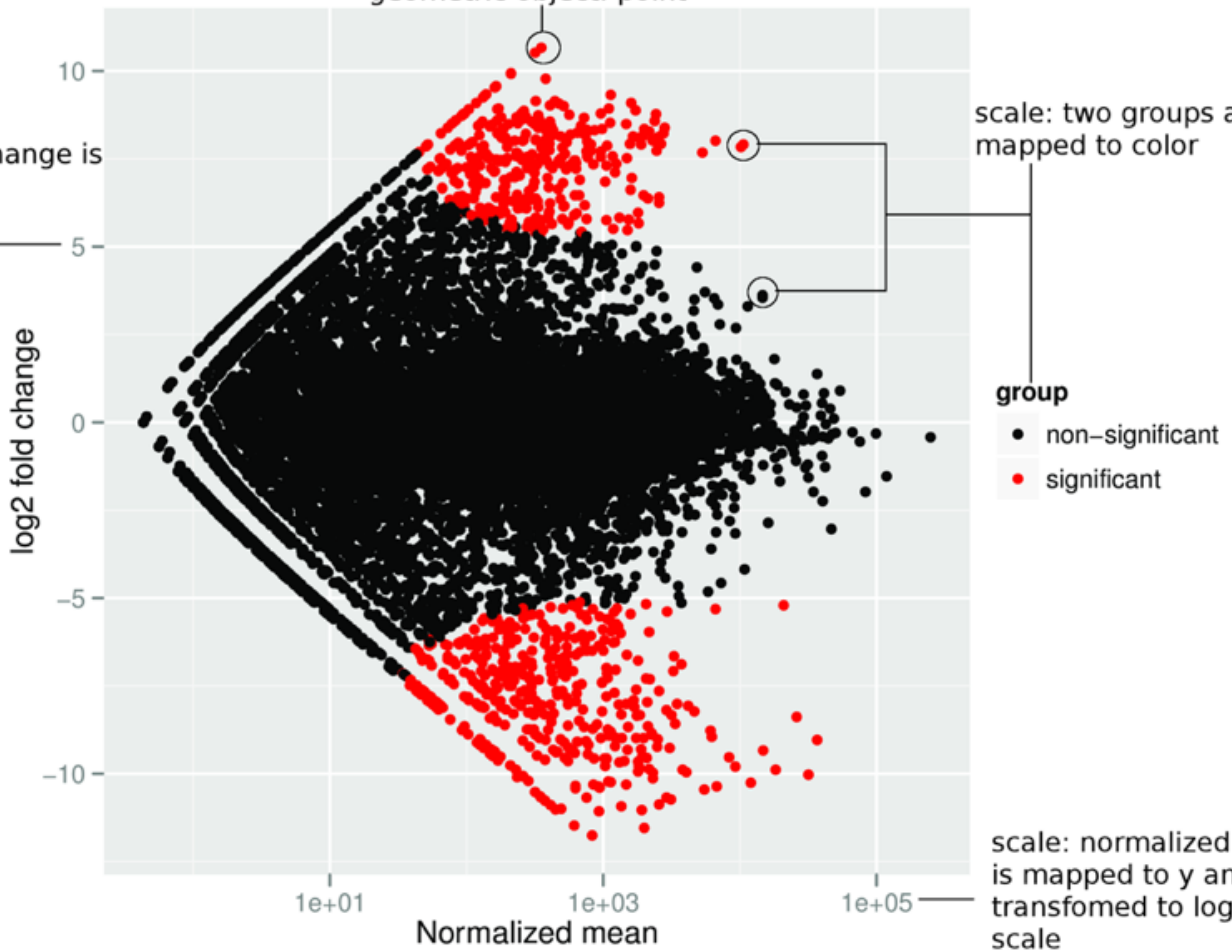
scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot

geometric object: point



scale: log2 fold change is mapped to y

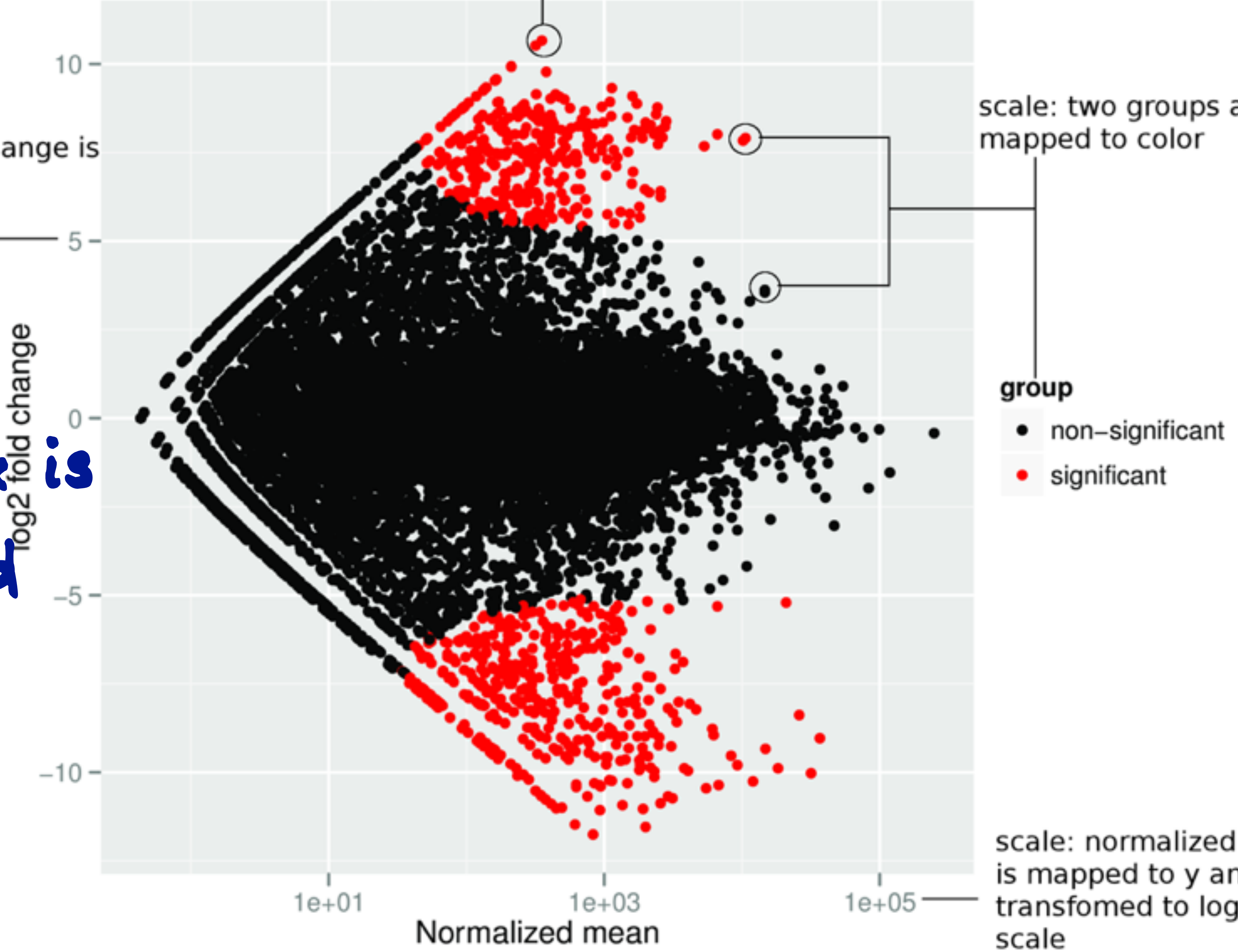
scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot

geometric object: point

scale: log2 fold change is mapped to y

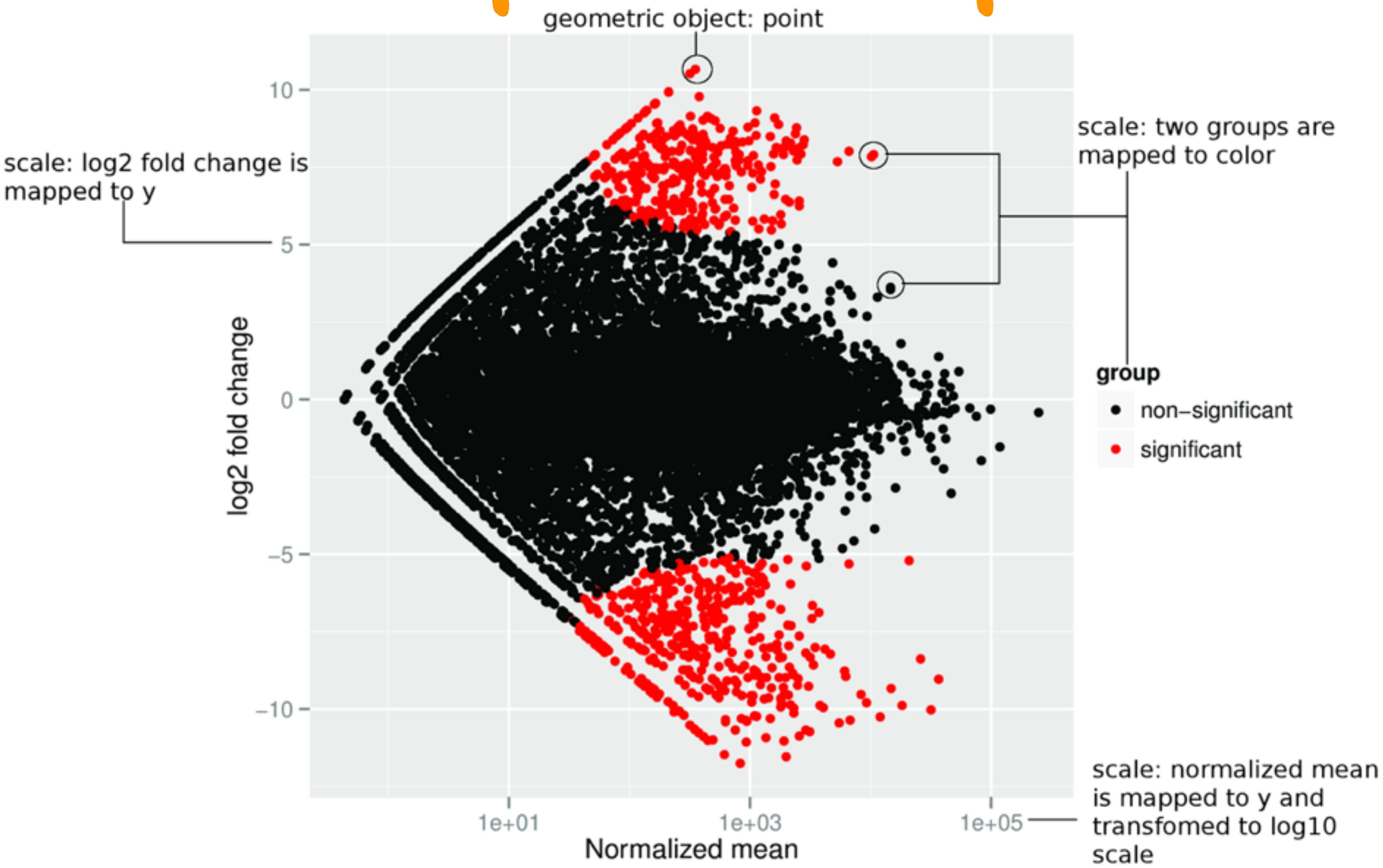


scale: two groups are mapped to color

SCALE=x is logged

scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot



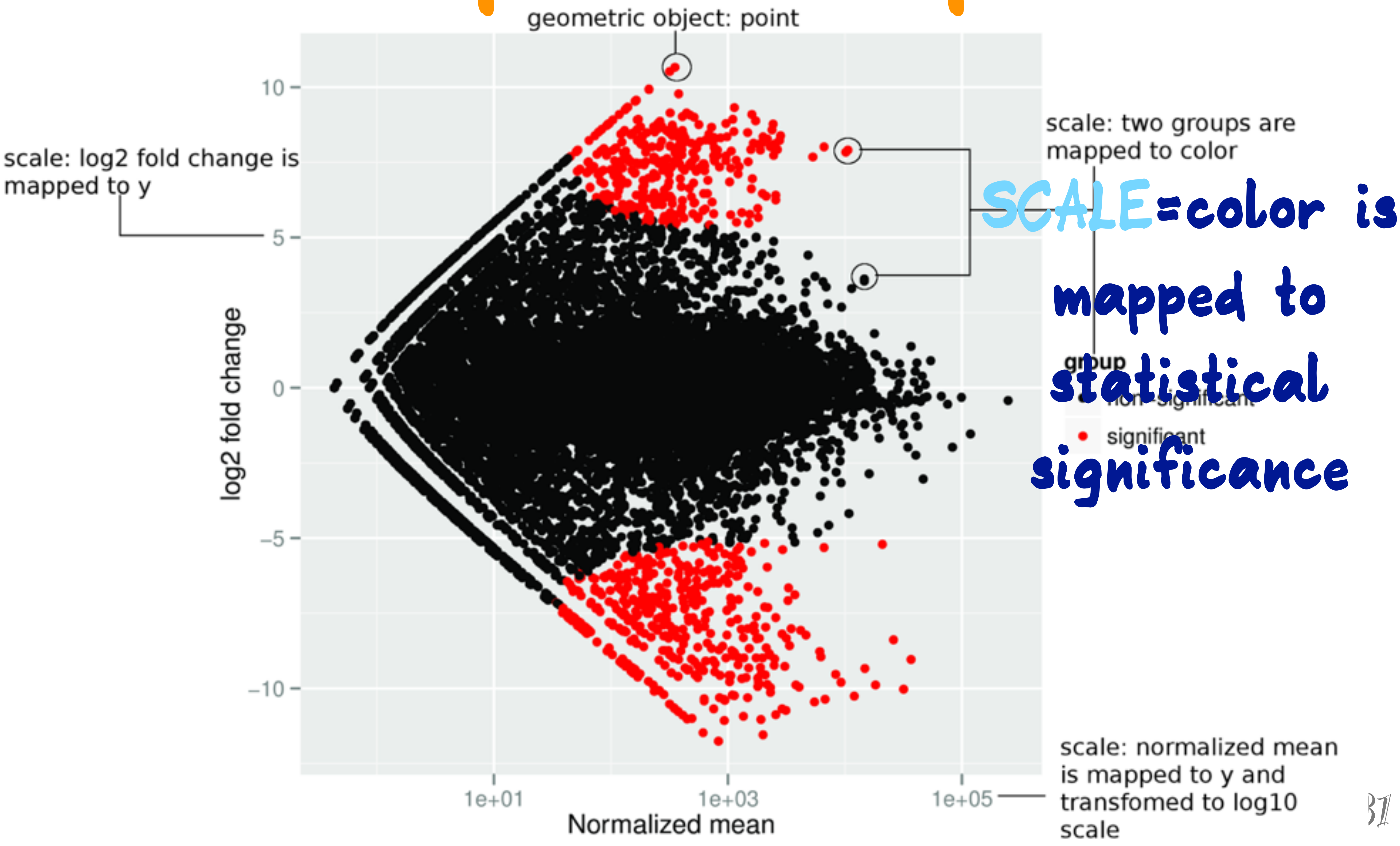
geometric object: point

scale: log2 fold change is mapped to y

scale: two groups are mapped to color

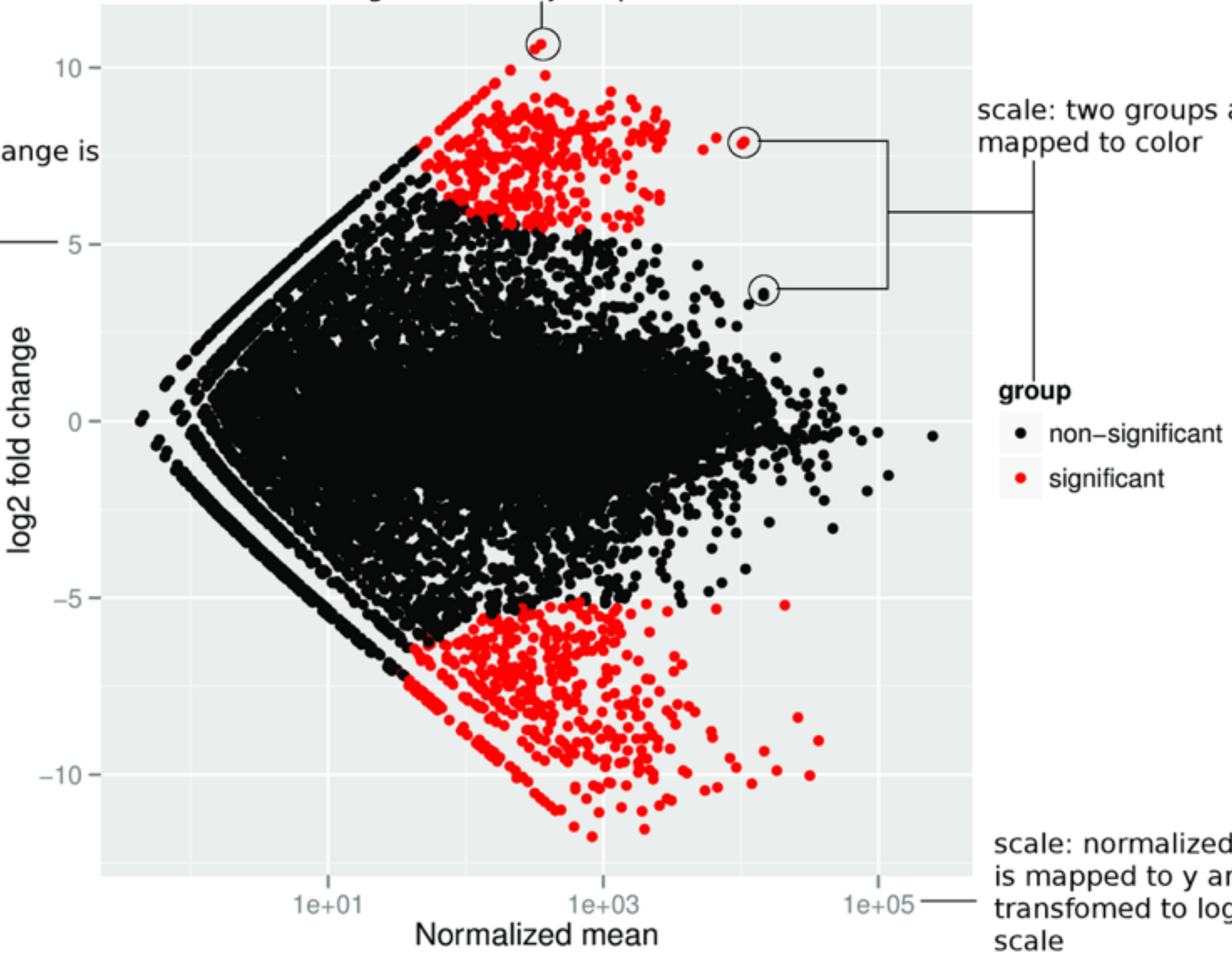
scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot



SCALE=color is mapped to statistical significance

Example: MA plot



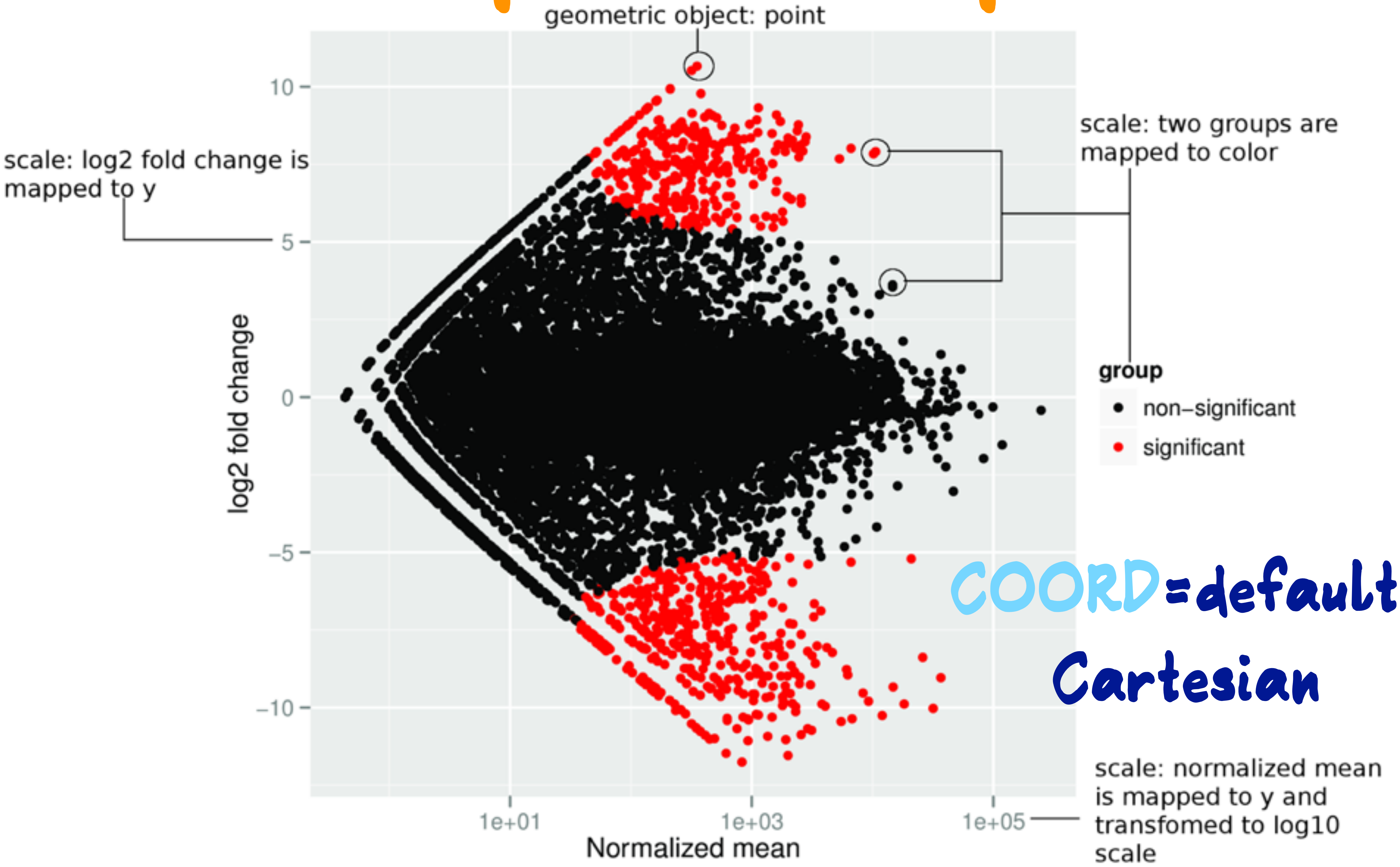
geometric object: point

scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot



geometric object: point

scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

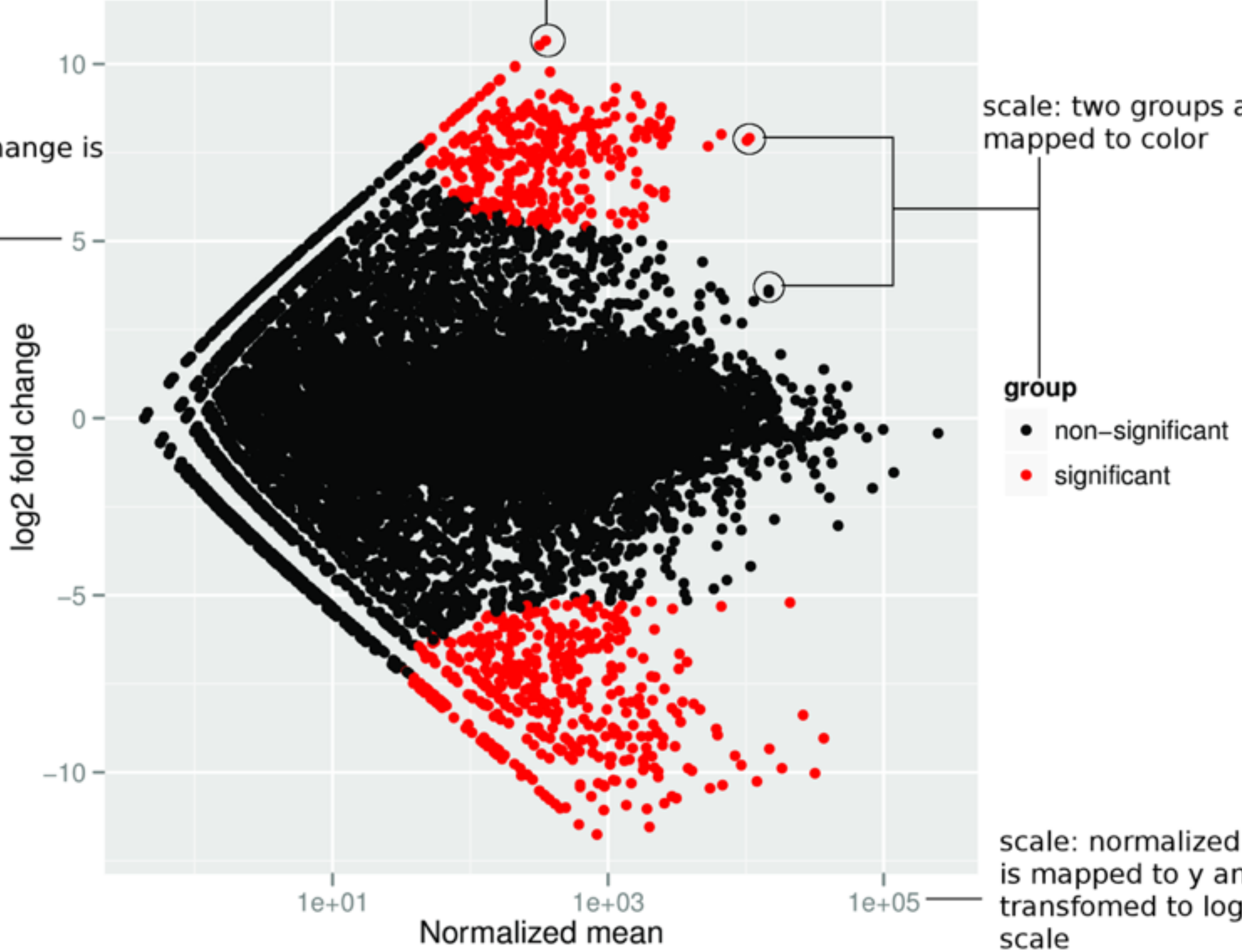
COORD=default, Cartesian

Example: MA plot

geometric object: point

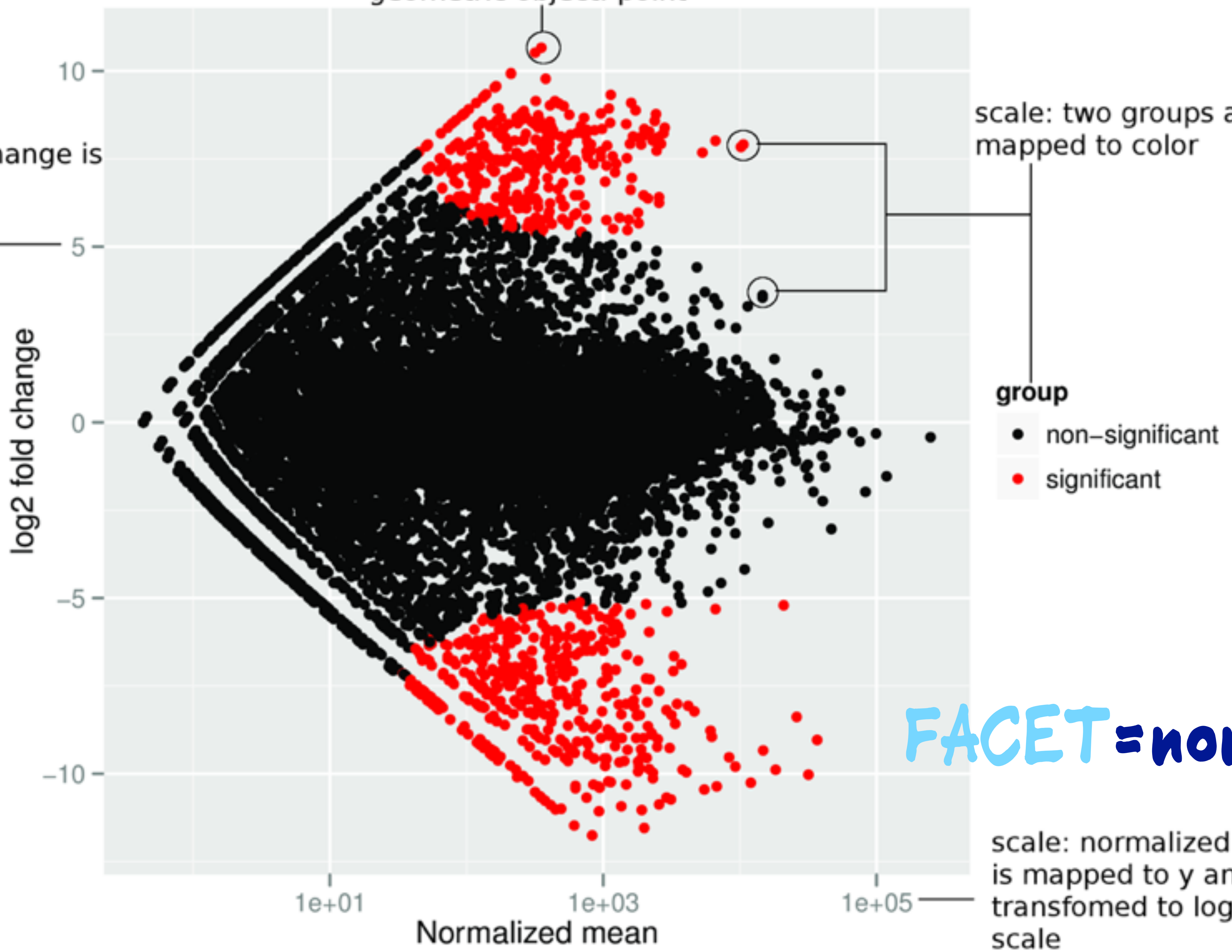
scale: log2 fold change is mapped to y

scale: two groups are mapped to color



scale: normalized mean is mapped to y and transfomed to log10 scale

Example: MA plot



geometric object: point

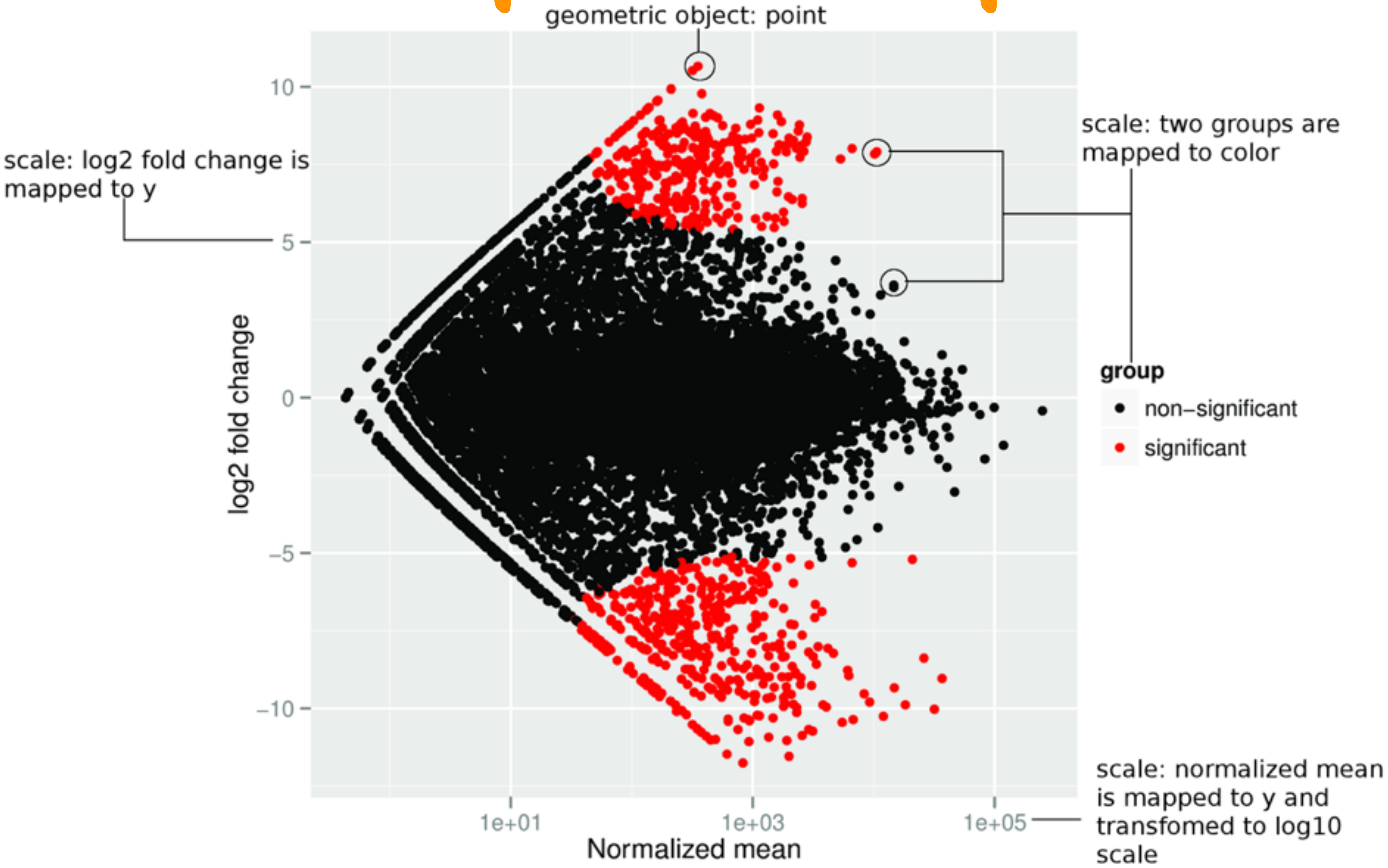
scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

FACET=none

Example: MA plot

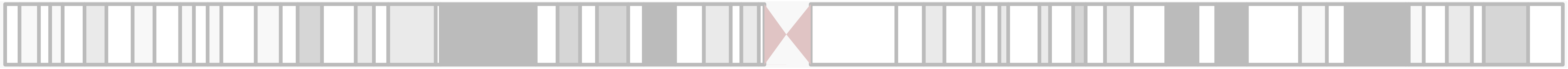


geometric object: point

scale: log2 fold change is mapped to y

scale: two groups are mapped to color

scale: normalized mean is mapped to y and transfomed to log10 scale

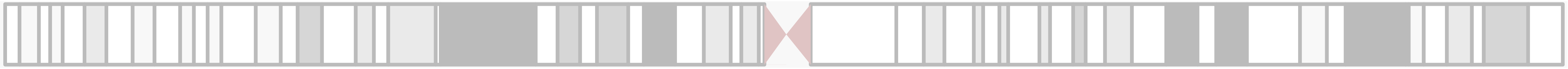


Example: MA plot

```
qplot(baseMean, log2FoldChange,
  data = res, geom = "point",
  xlab = "Normalized mean",
  ylab = "log2 fold change",
  xlim = c(0, 10000),
  color = group) +
  scale_x_log10() +
  scale_color_manual(
  values = c("black", "red"))
```

What's different?

- Genomic data has interval context
- Several common geoms used in standard plots, not in current grammar
- Additional transformations common
- Lining up of multiple data plots, especially against genome

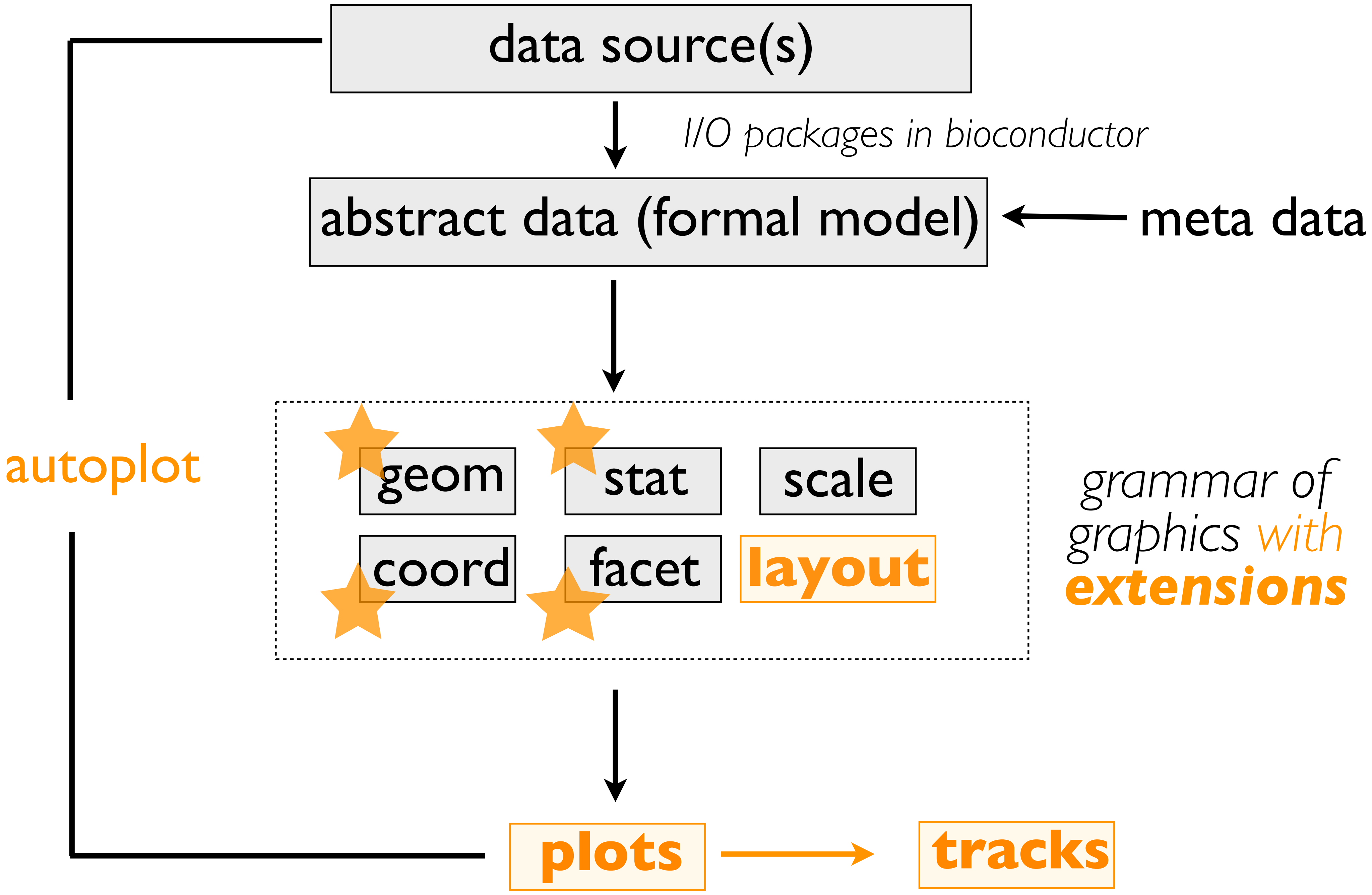


What's different?

No	seqnames	ranges	strand	tx_id	exon_id
1	chrX	[48242968, 48243005]	+	35775	132624
2	chrX	[48243475, 48243563]	+	35775	132625
3	chrX	[48244003, 48244117]	+	35775	132626
4	chrX	[48244794, 48244889]	+	35775	132627
5	chrX	[48246753, 48246802]	+	35775	132628
...	...	...	...	... ...	...
26	chrX	[48270193, 48270307]	-	35778	132637
27	chrX	[48269421, 48269516]	-	35778	132636
28	chrX	[48267508, 48267557]	-	35778	132635
29	chrX	[48262894, 48262998]	-	35778	132633
30	chrX	[48261524, 48262111]	-	35778	132632

DATA: **Genomic ranges**

Extensions



data source(s)

I/O packages in bioconductor

abstract data (formal model)

meta data

geom

stat

scale

coord

facet

layout

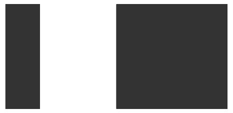
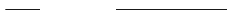
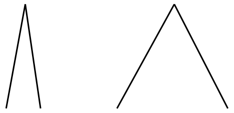
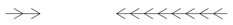
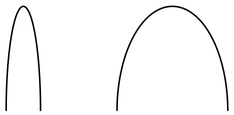
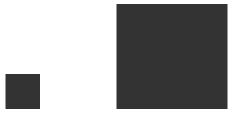
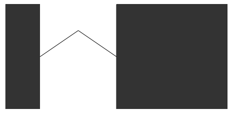
grammar of graphics with ***extensions***

autoplot

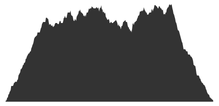
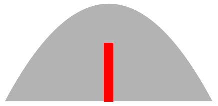
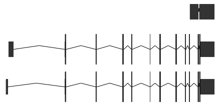
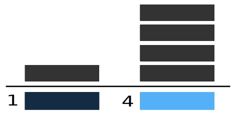
plots

tracks

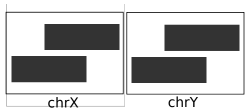
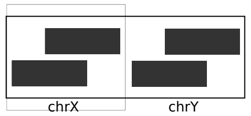
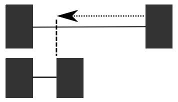
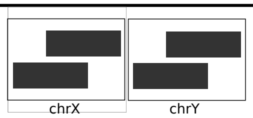
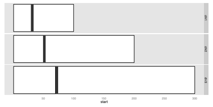
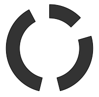
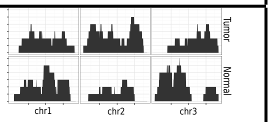
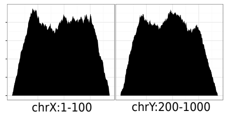
Extensions

Comp	name	usage	icon
geom	geom_rect	rectangle	
	geom_segment	segment	
	geom_chevron	chevron	 ★
	geom_arrow	arrow	 ★
	geom_arch	arches	 ★
	geom_bar	bar	
	geom_alignment	alignment (gene)	 ★

Extensions

stat	stat_coverage	coverage (of reads)	
	stat_mismatch	mismatch pileup for alignments	
	stat_aggregate	aggregate in sliding window	
	stat_stepping	avoid overplotting	
	stat_gene	consider gene structure	
	stat_table	tabulate ranges	
	stat_identity	no change	

Extensions

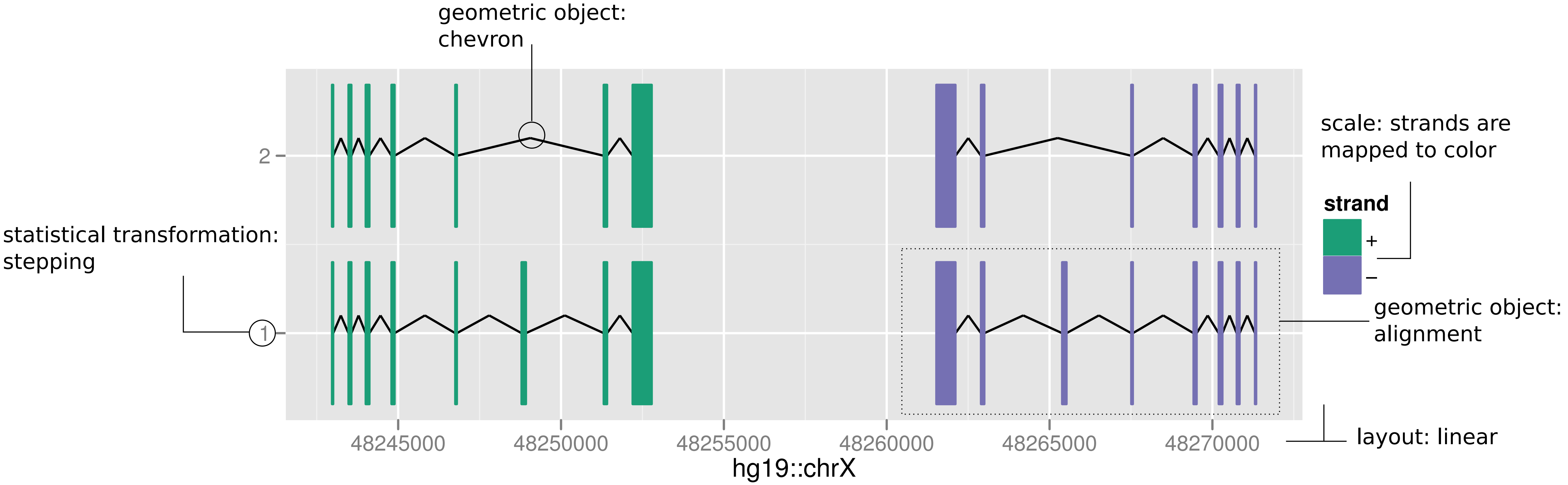
coord	linear	ggplot2 linear but facet by chromosome		
	genome	put everything on genominc coordinates		★
	truncate gaps	compact view by shrinking gaps		★
layout	track	stacked tracks		★
	karyogram	karyogram display		★
	circle	circular		★
faceting	formula	facet by formula		
	ranges	facet by ranges		★

Extensions

autoplot

Tries, and does a jolly good job, of recognizing the data object to be plotted, and how it should be displayed.

Example



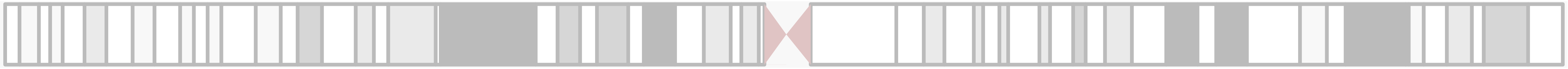
geometric object:
chevron

scale: strands are
mapped to color

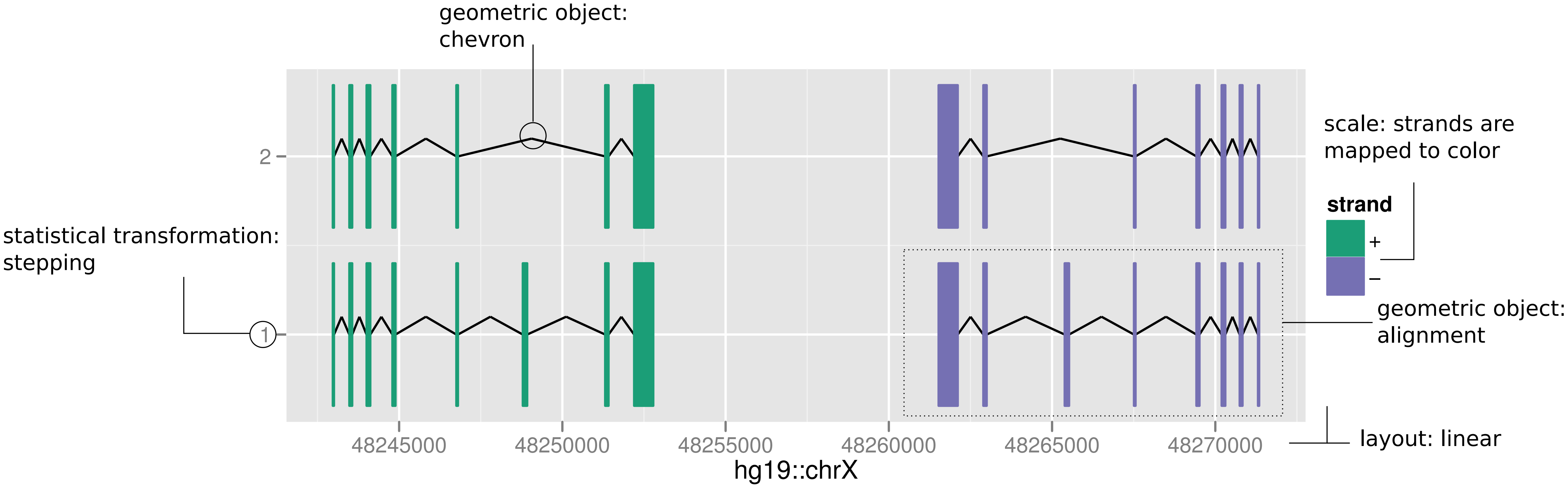
statistical transformation:
stepping

geometric object:
alignment

layout: linear



Example



geometric object: chevron

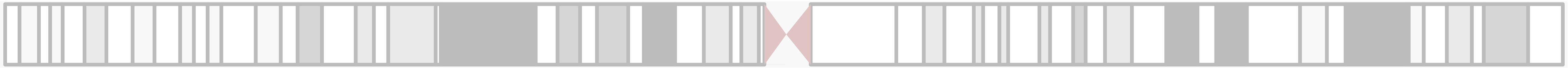
scale: strands are mapped to color

statistical transformation: stepping

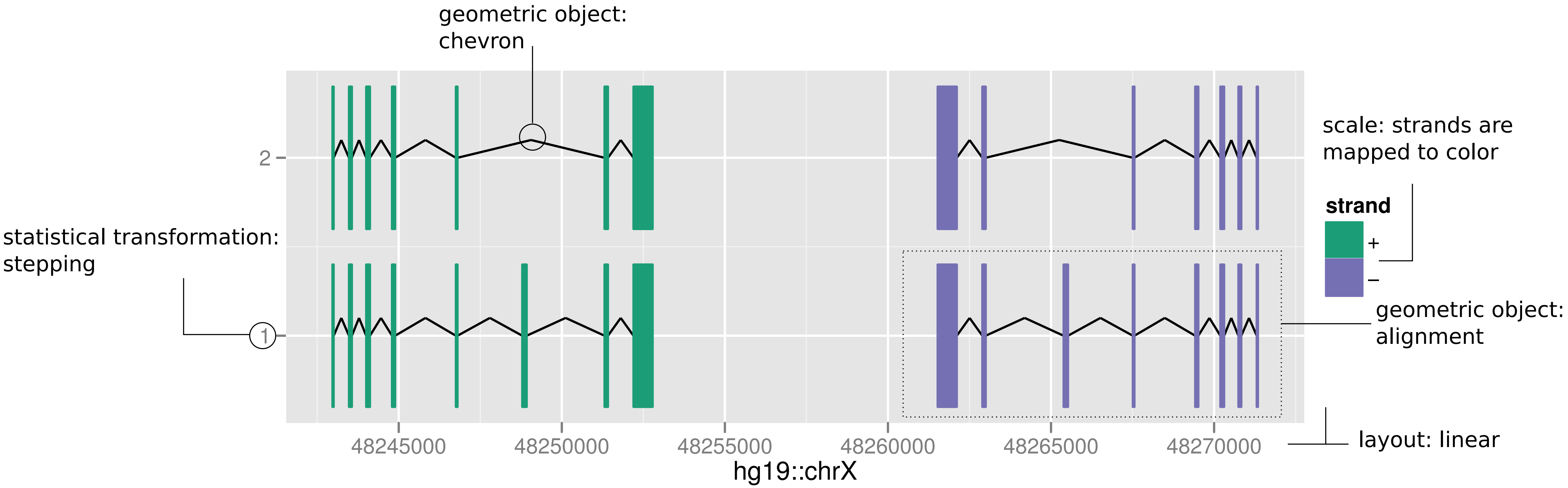
geometric object: alignment

layout: linear

DATA=GRangesList Object



Example



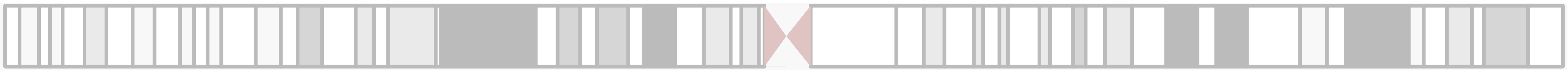
geometric object: chevron

statistical transformation: stepping

scale: strands are mapped to color

geometric object: alignment

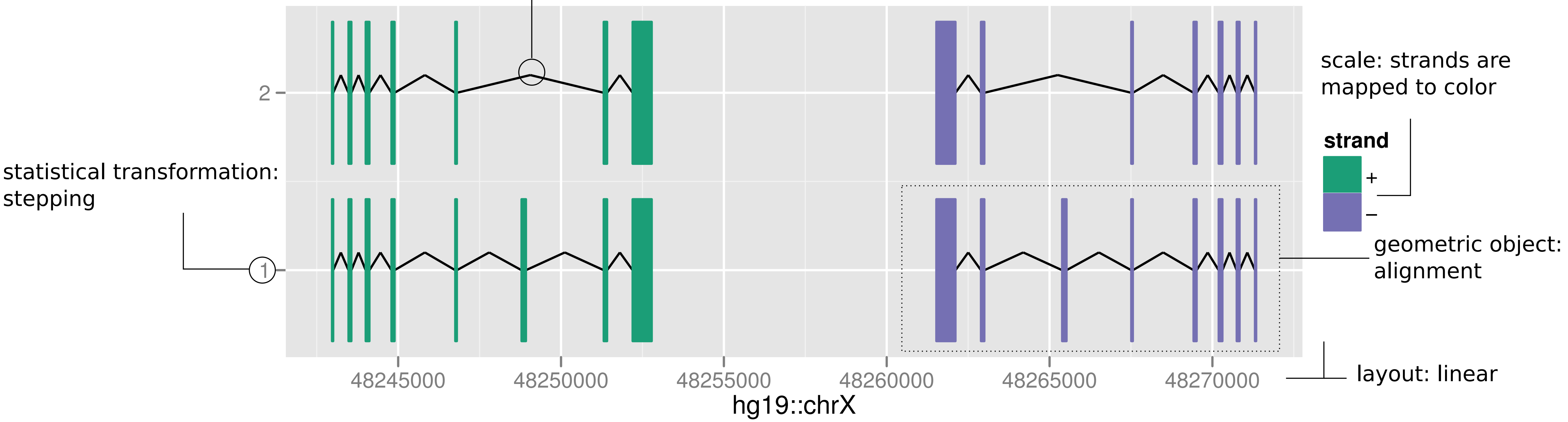
layout: linear



Example

GEOM=alignment, chevron

geometric object: chevron

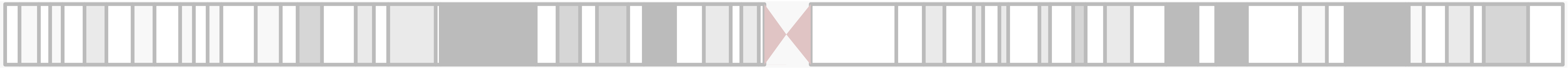


statistical transformation: stepping

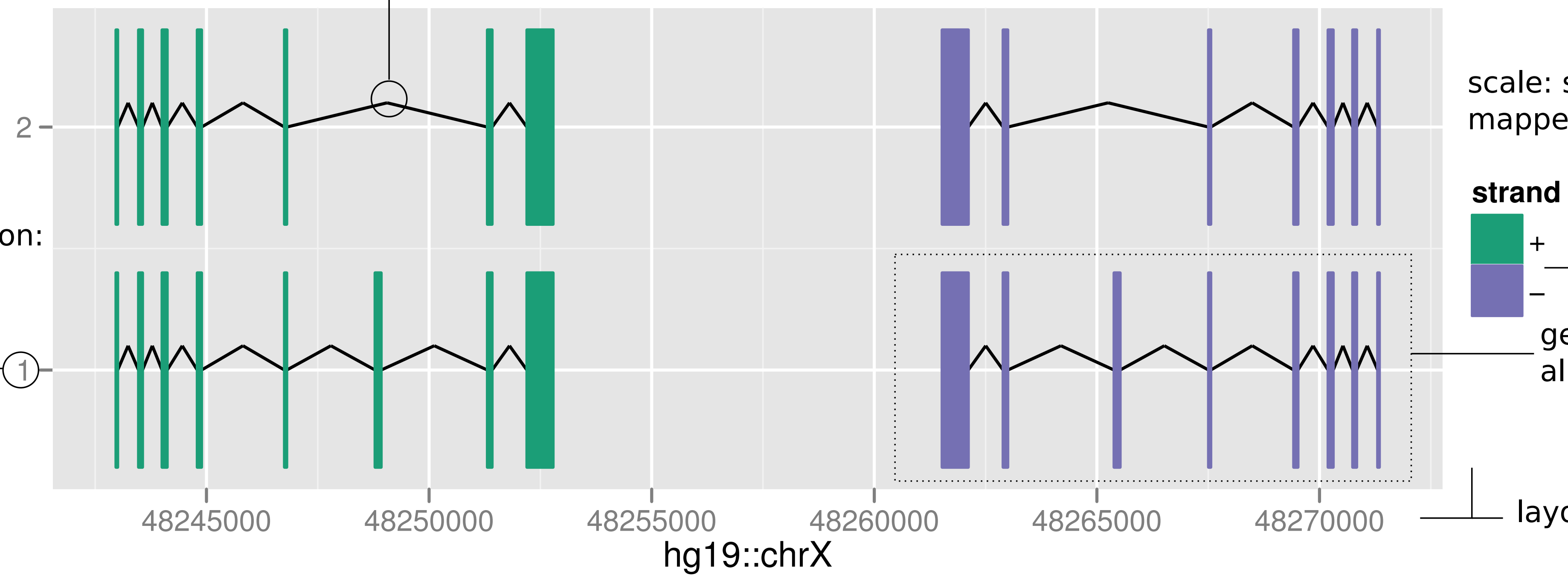
scale: strands are mapped to color

geometric object: alignment

layout: linear



Example



geometric object: chevron

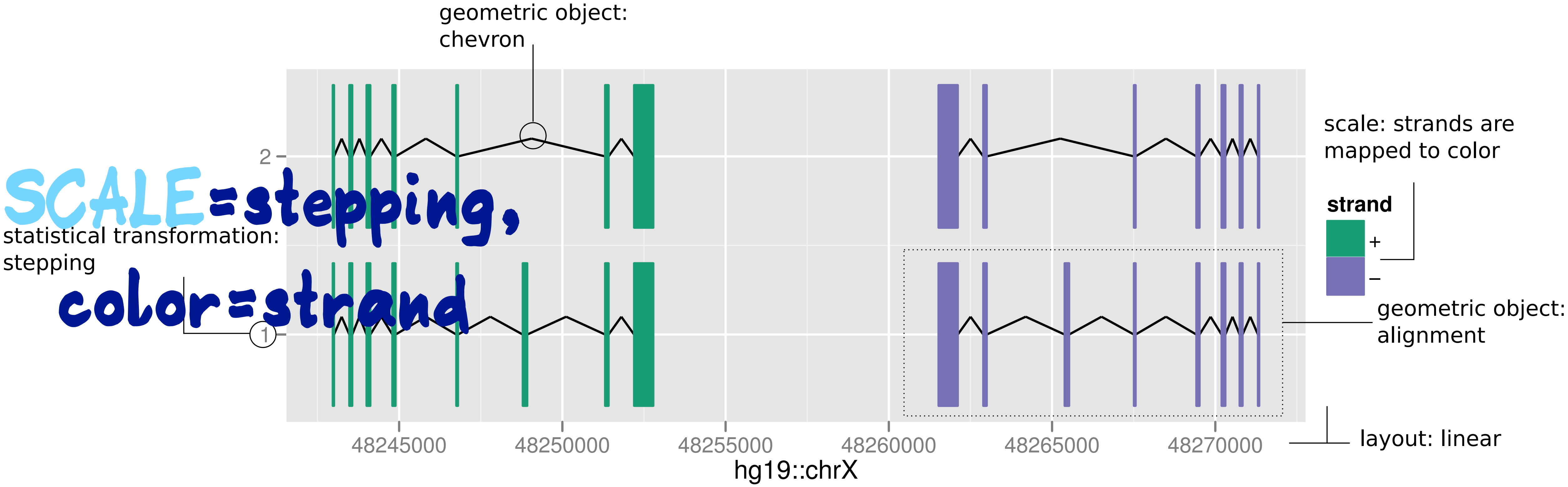
scale: strands are mapped to color

statistical transformation: stepping

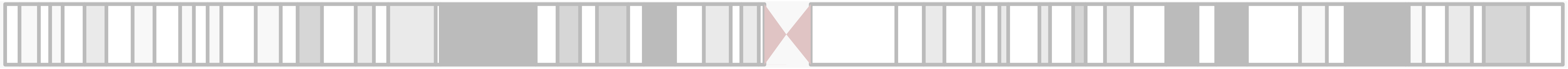
geometric object: alignment

layout: linear

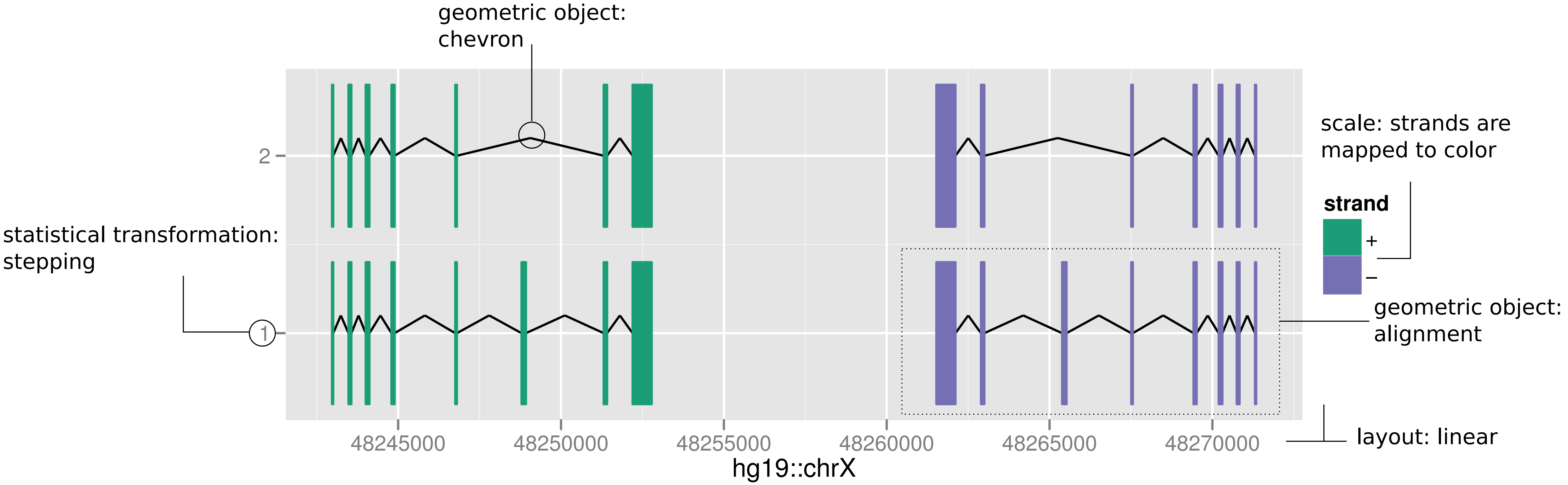
Example



SCALE=stepping,
color=strand



Example



geometric object: chevron

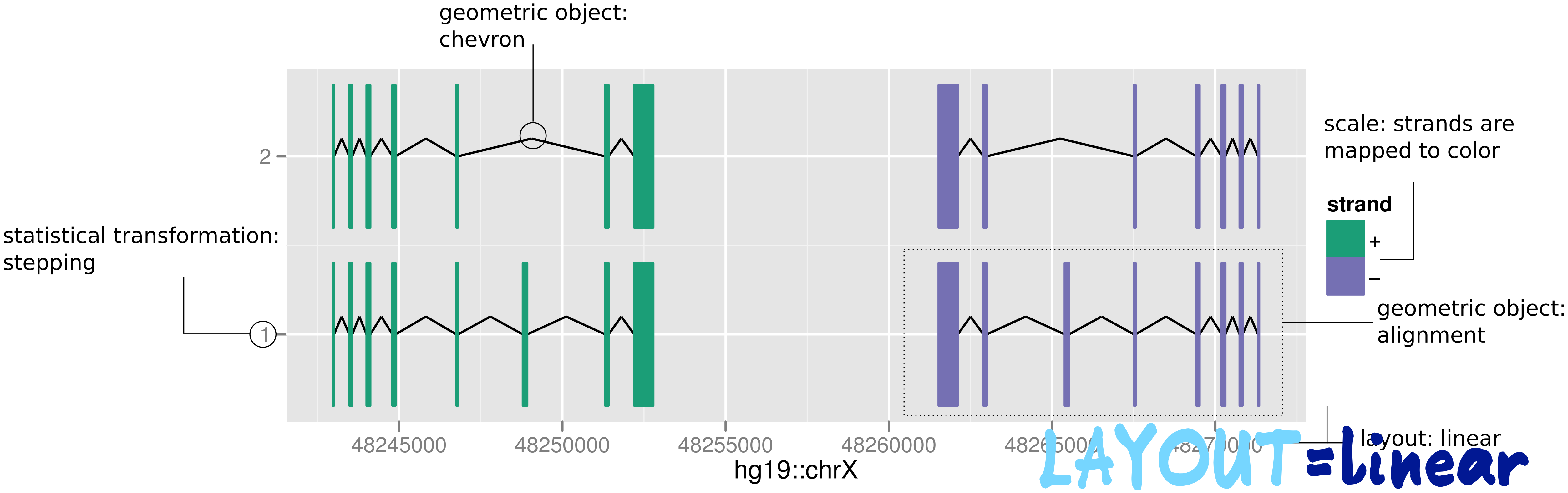
statistical transformation: stepping

scale: strands are mapped to color

geometric object: alignment

layout: linear

Example



geometric object: chevron

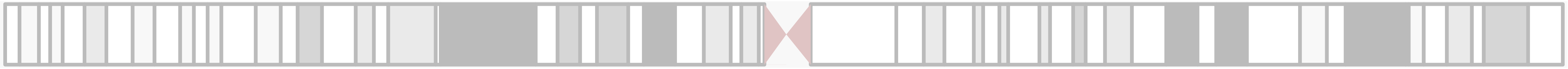
scale: strands are mapped to color

statistical transformation: stepping

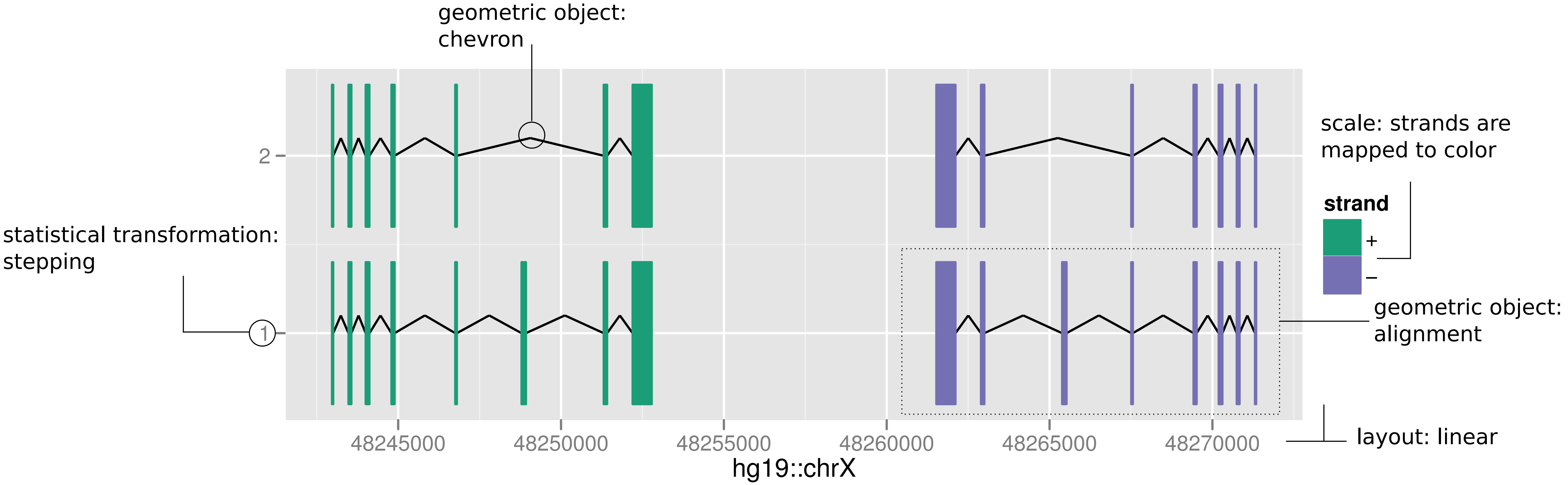
geometric object: alignment

layout: linear

LAYOUT=Linear



Example



geometric object: chevron

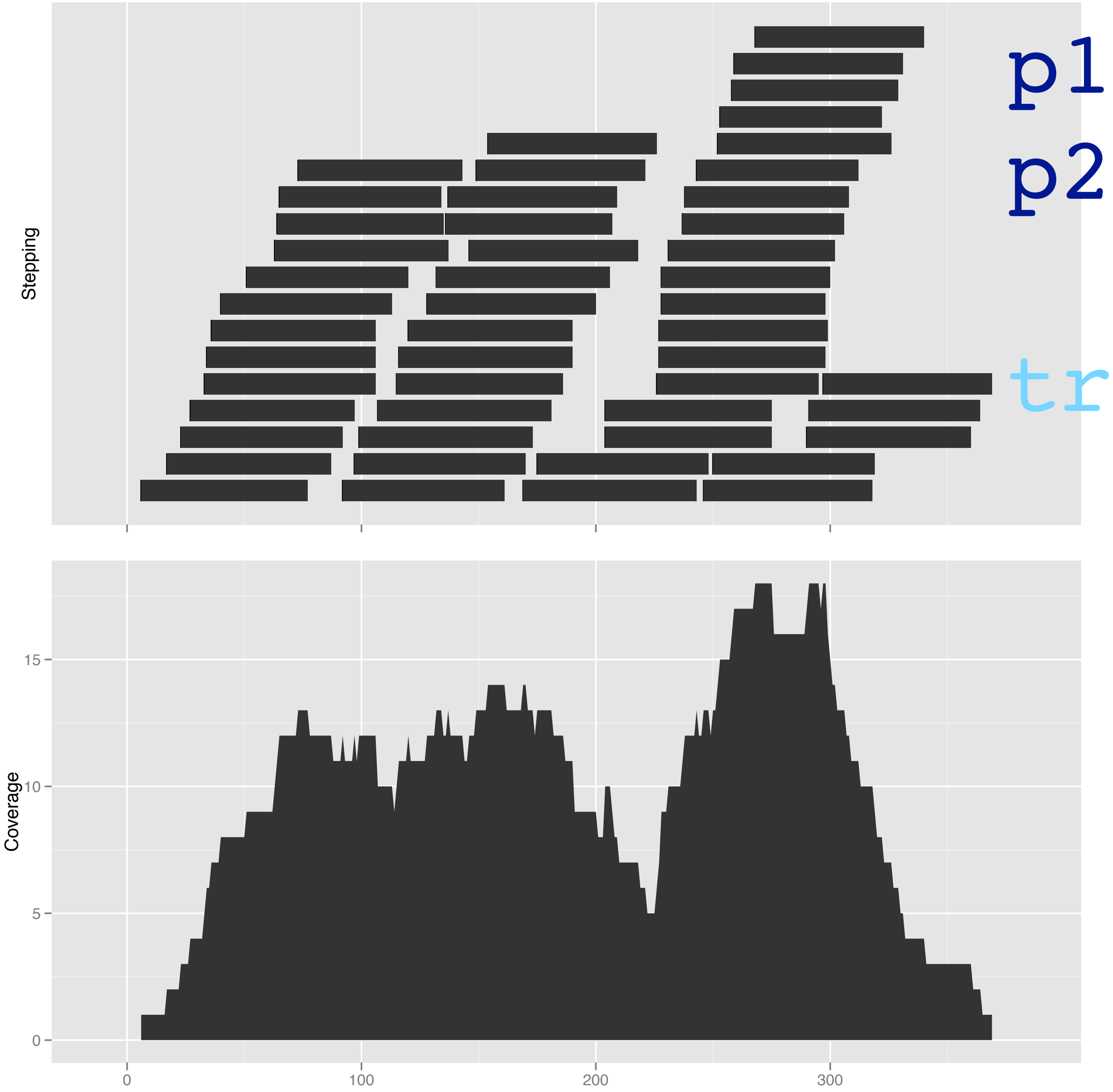
scale: strands are mapped to color

statistical transformation: stepping

geometric object: alignment

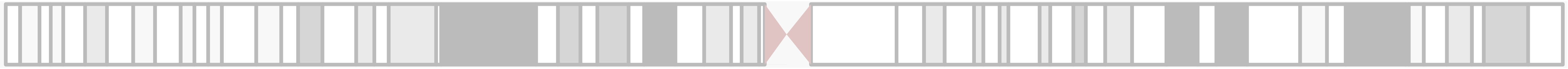
layout: linear

Examples

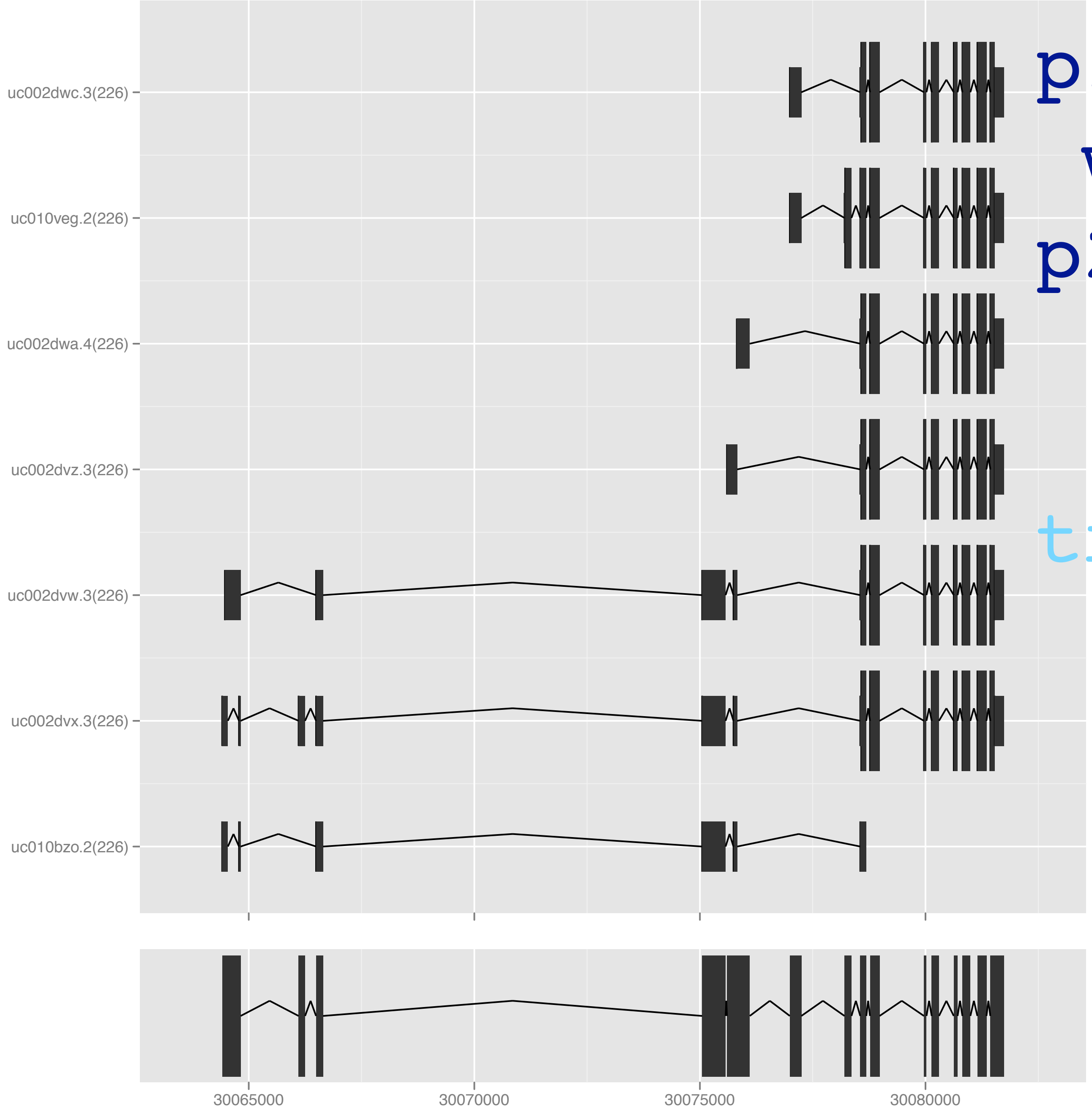


```
p1 <- autoplot(gr)
p2 <- autoplot(gr,
        stat = "coverage")
tracks(p1, p2)
```

- Examine short reads
- Stack them (top)
- Collapse into "density" (bottom)



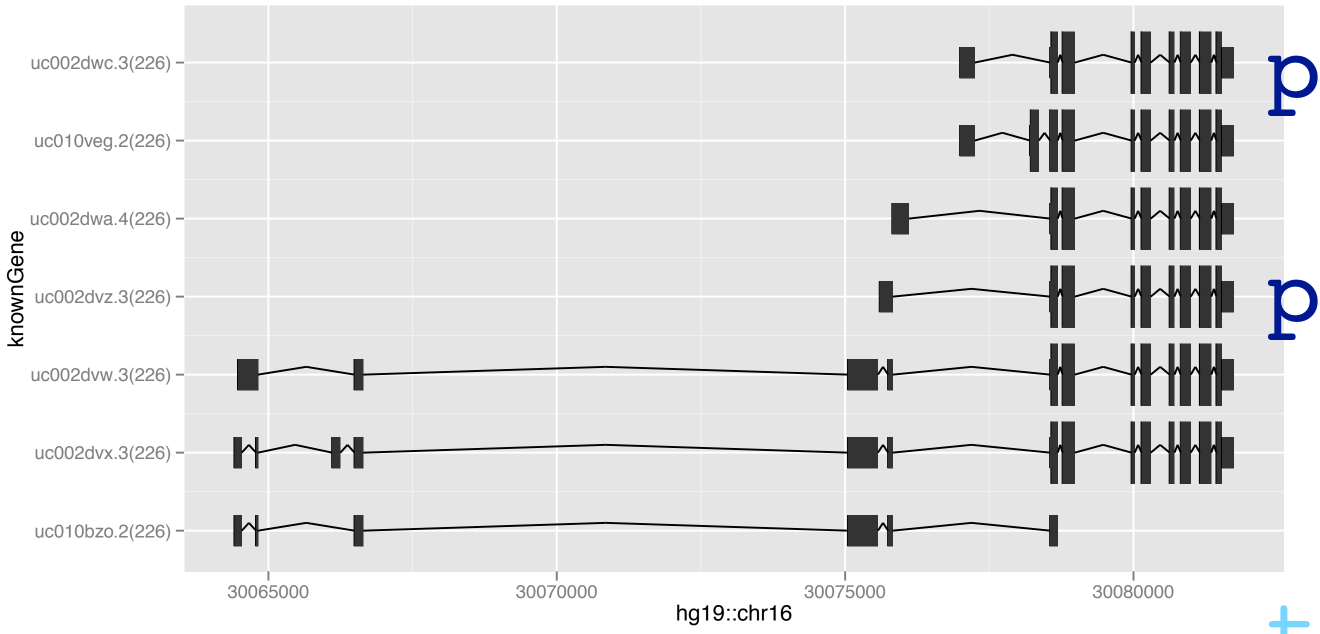
Examples

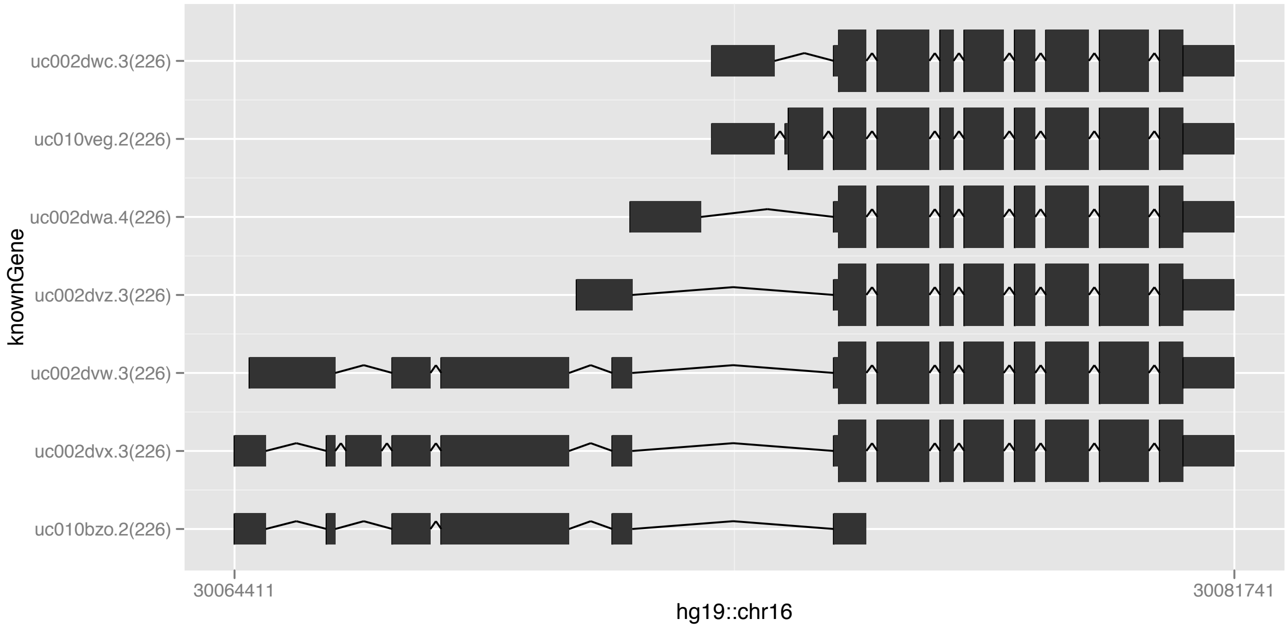


```
p1 <- autoplot(txdb,
  which = genesymbol["A"])
p2 <- autoplot(txdb,
   which = genesymbol["A"],
   stat = "reduce")
tracks(p1, p2,
   heights = c(4, 1))
```

- Compare transcripts
- Reduce all to one

Examples

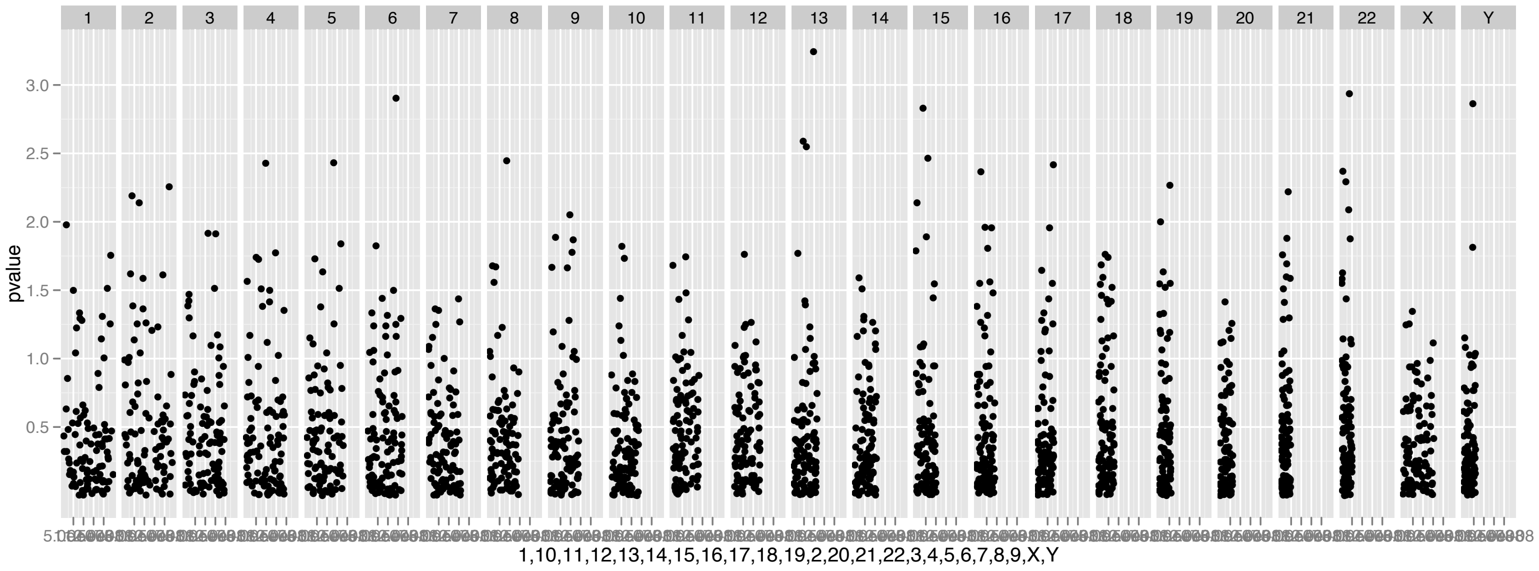




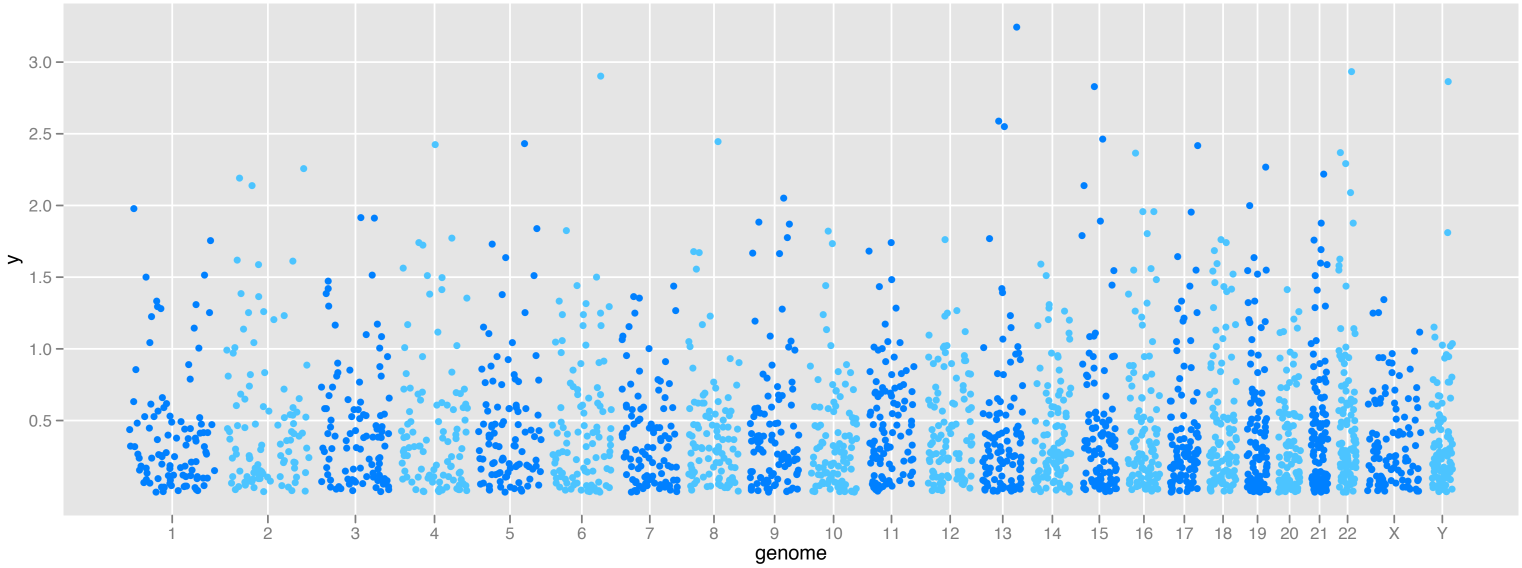
```
p1 <- autoplot(txdb,
 which = genesymbol["A"])
p2 <- autoplot(txdb,
  which = genesymbol["A"],
  truncate.gaps = TRUE)
tracks(p1, p2,
  heights = c(4, 4))
```

Focus on exons

Examples



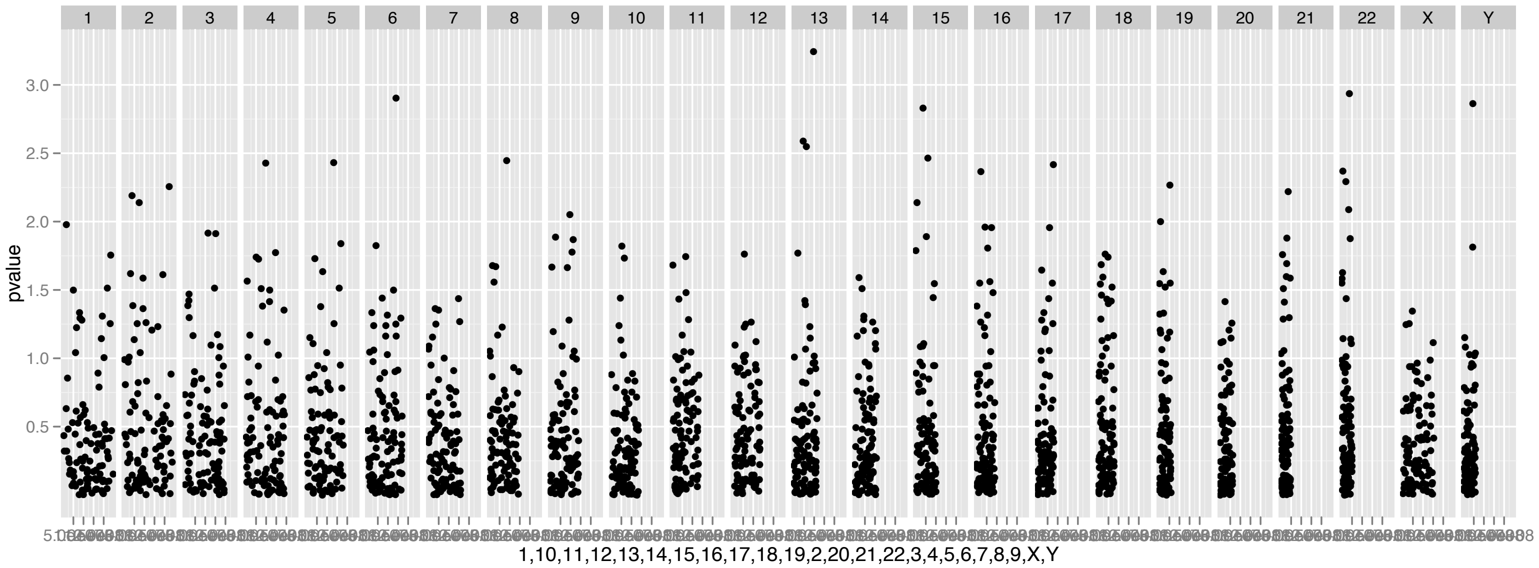
```
p1 <- autoplot(gr.snp,
   geom = "point",
   aes(y = pvalue))
```



```
p2 <- plotGrandLinear(
   gr.snp,
   aes(y = pvalue))
```

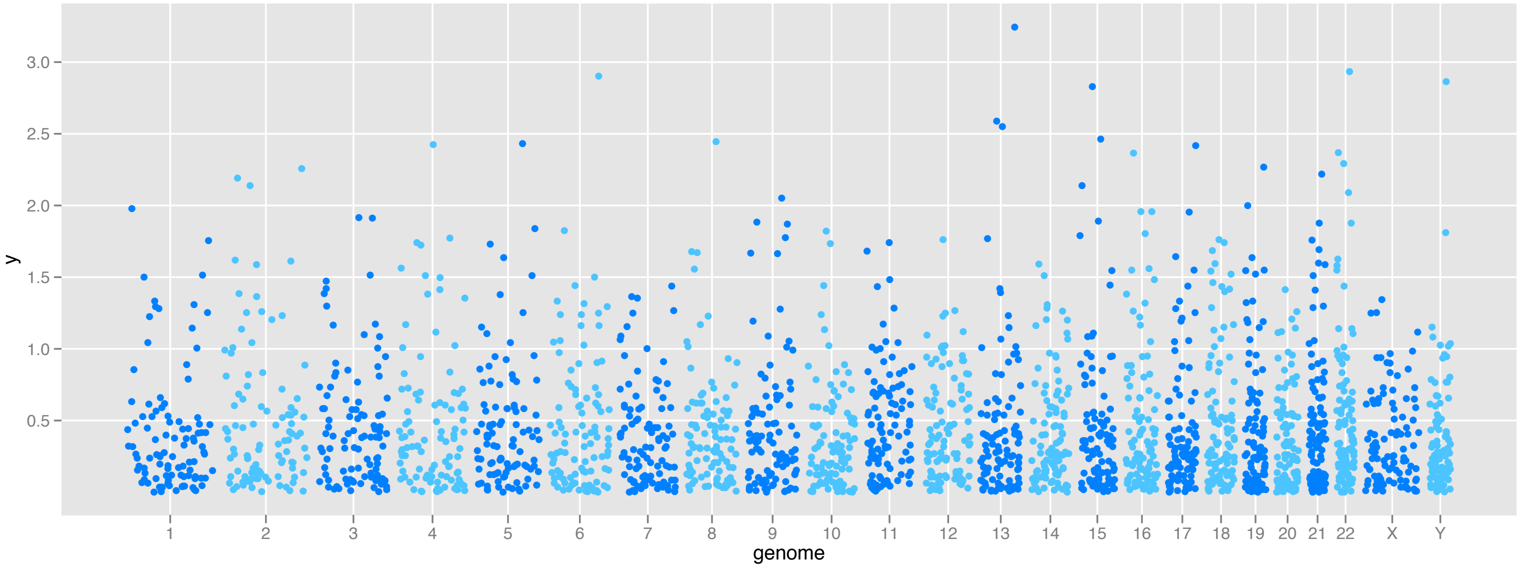
- Manhattan plot: features plotted against genomic position

Examples



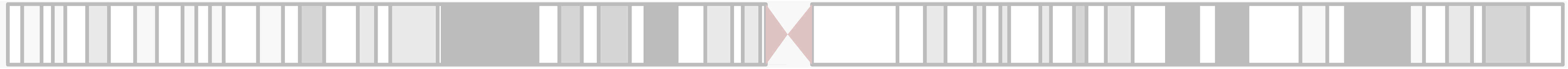
```
p1 <- autoplot(gr.snp,
   geom = "point",
   aes(y = pvalue))
```

Facets by chromosome #

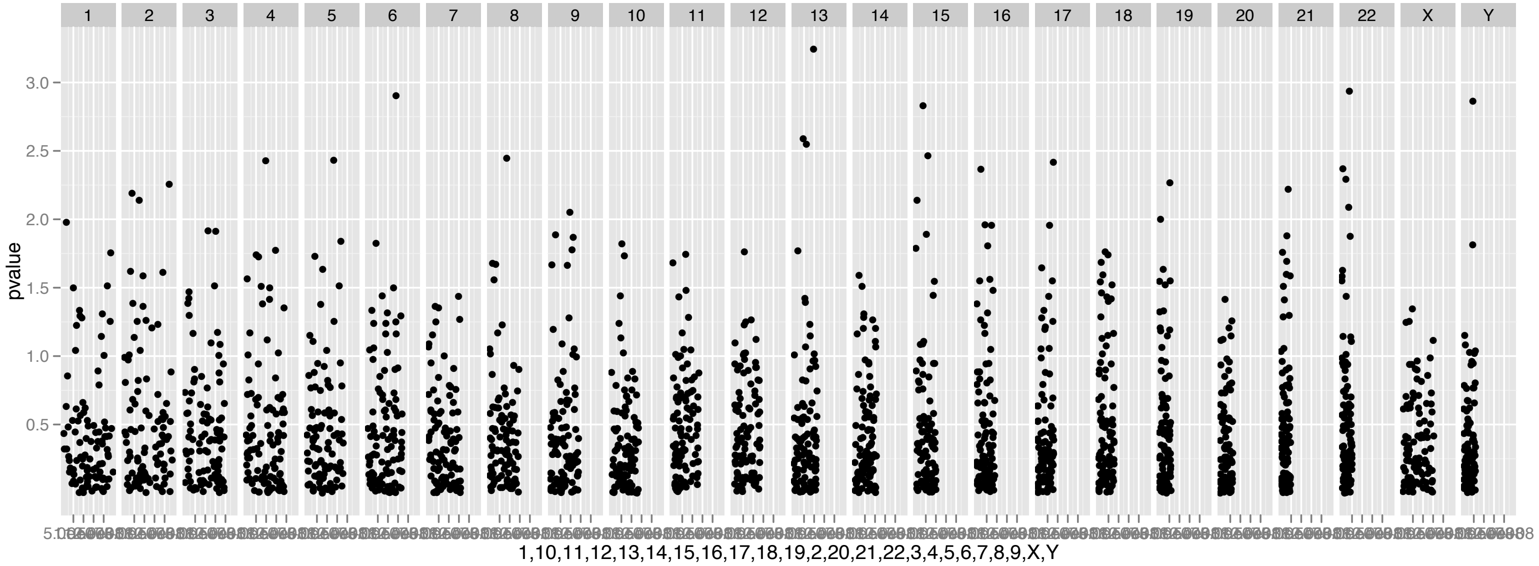


```
p2 <- plotGrandLinear(
   gr.snp,
   aes(y = pvalue))
```

- Manhattan plot: features plotted against genomic position

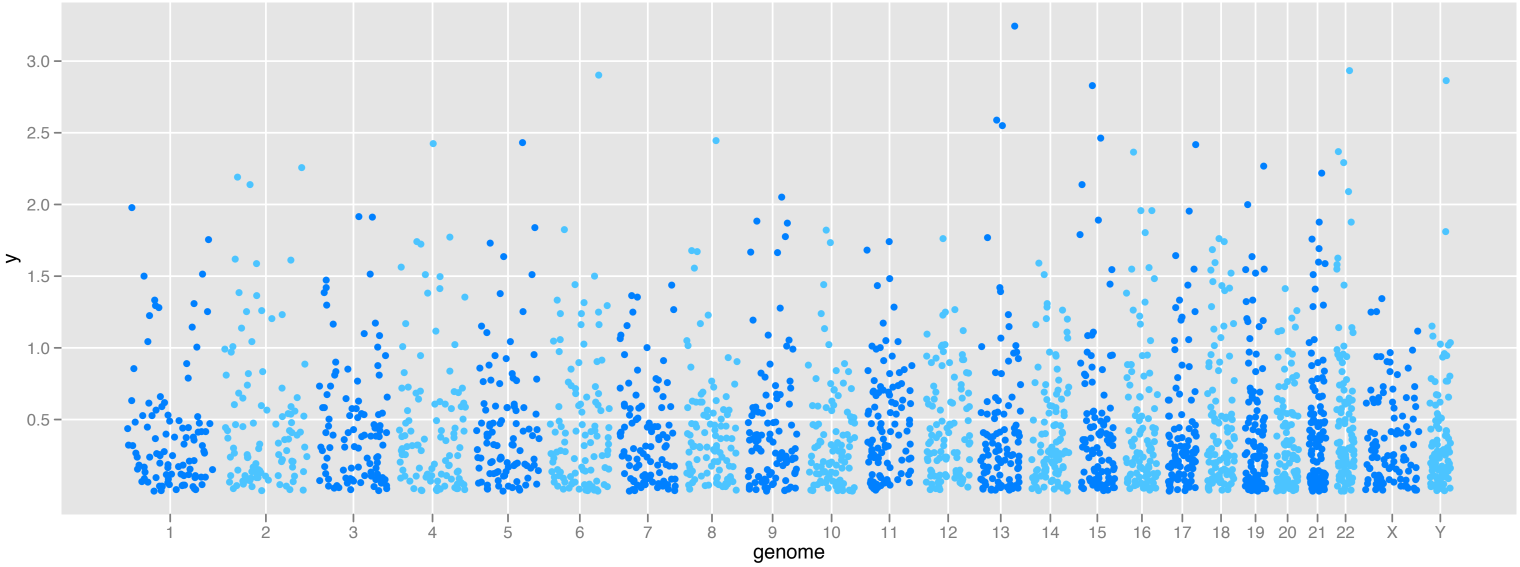


Examples



```
p1 <- autoplot(gr.snp,
   geom = "point",
   aes(y = pvalue))
```

Facets by chromosome #

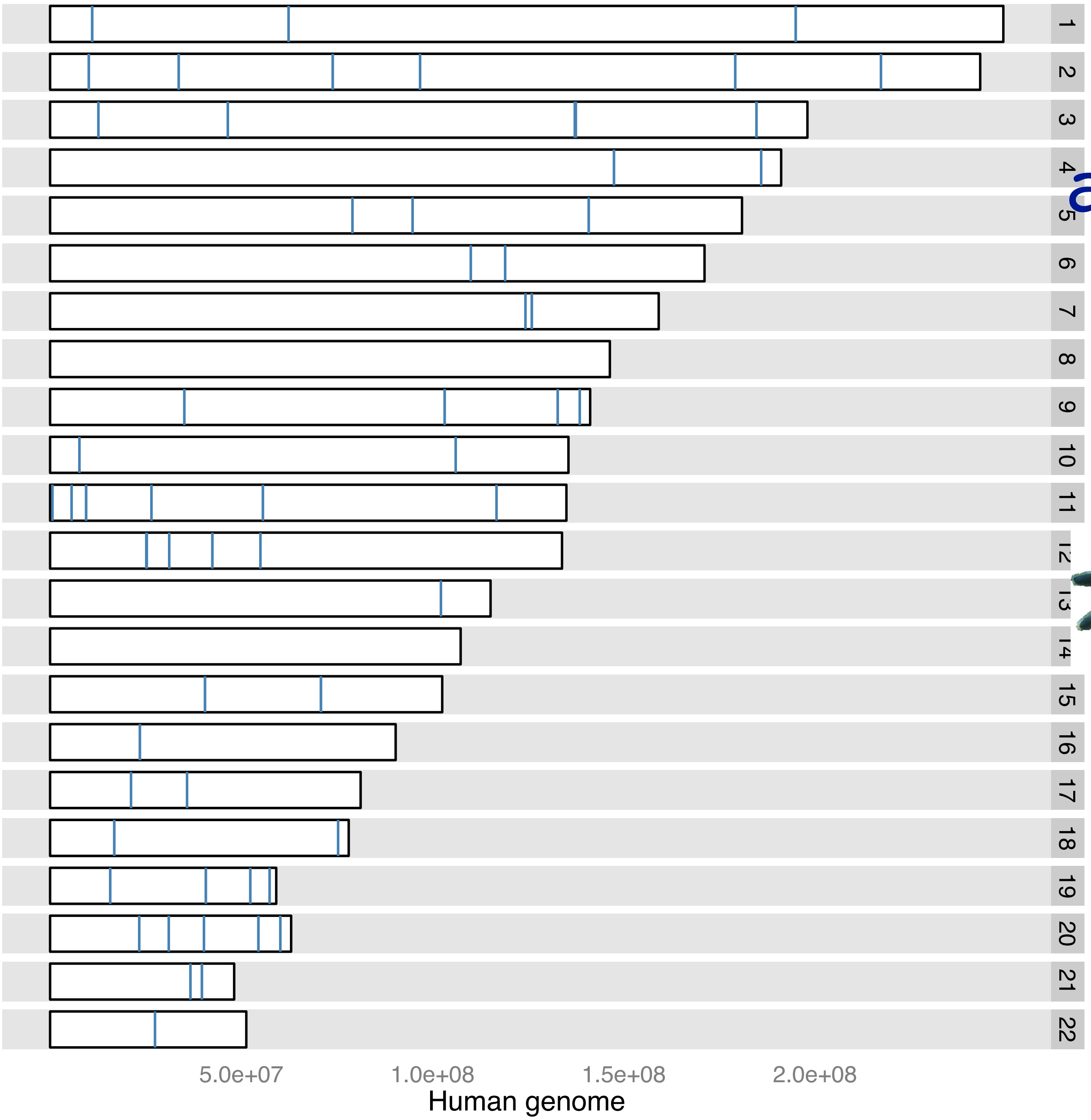


```
p2 <- plotGrandLinear(
   gr.snp,
   aes(y = pvalue))
```

Turns chromosome # into numerical scale

- Manhattan plot: features plotted against genomic position

Examples



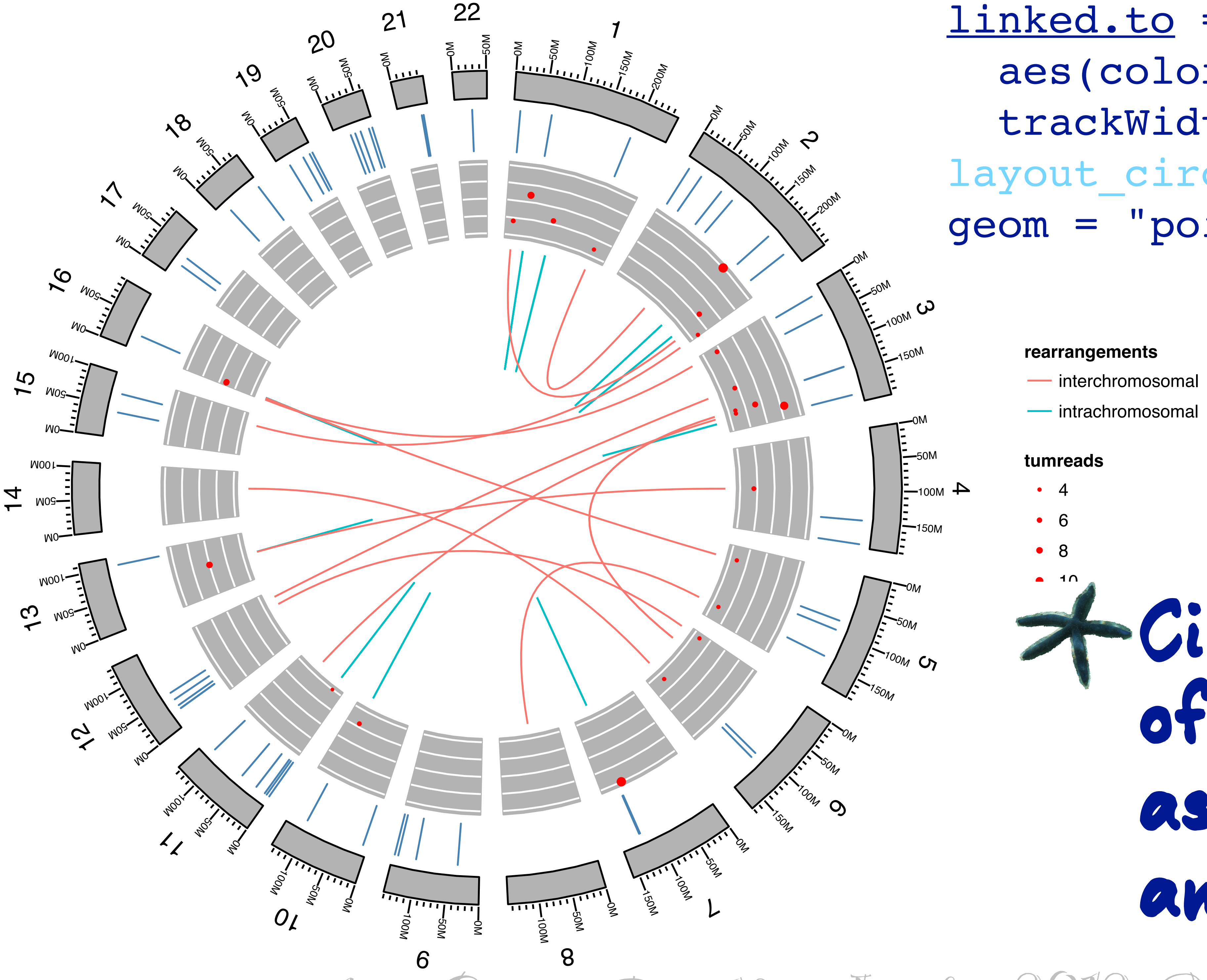
GRanges

```
autoplot(gr,
  layout = "karyogram",
  color = "blue")
```

Karyogram, highlight locations corresponding to some data feature

Examples

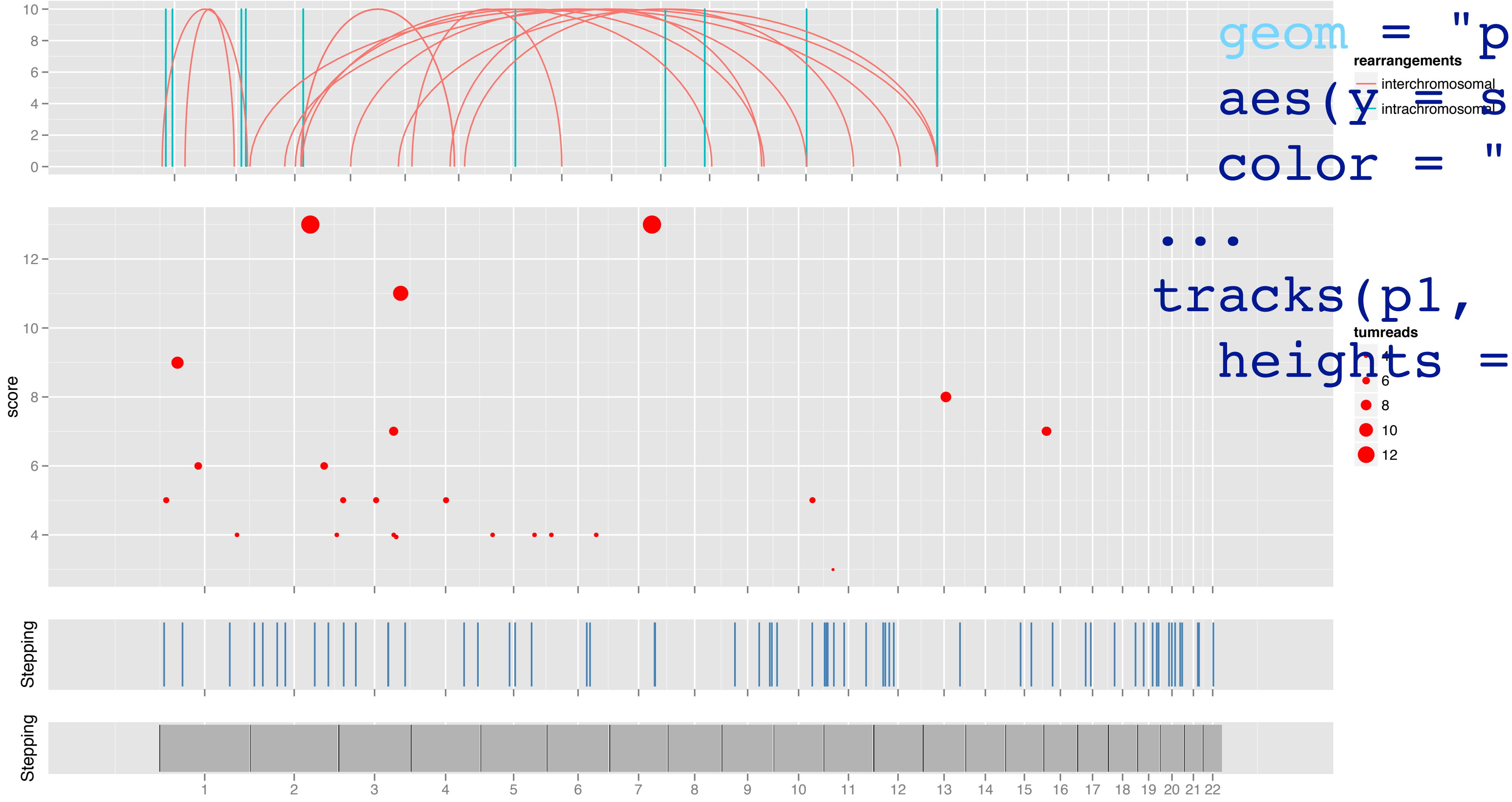
```
ggplot() +
  layout_circle(gr1,
    geom = "link",
  linked.to = "to.gr",
    aes(color = rearrangement),
    trackWidth = 1, radius = 10) +
  layout_circle(gr2,
  geom = "point", ...) + ...
```



Circular layout of genome with associated data, and connections

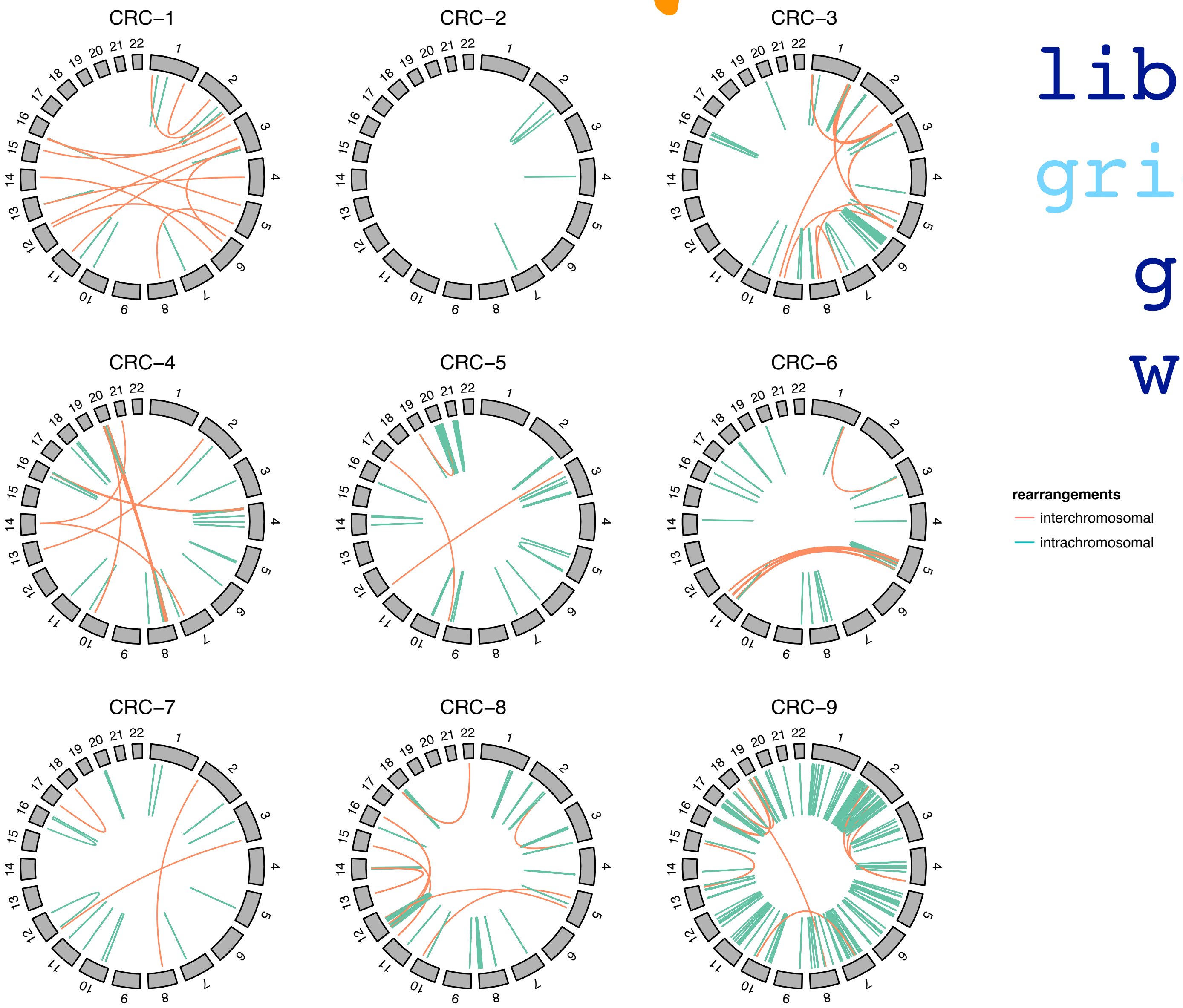
Examples

```
p1 <- autoplot(gr1,
   geom = "arch",
   aes(color = rearrangement),
   coord = "genome")
p2 <- autoplot(gr2,
   geom = "point",
   aes(y = score, size = tumreads),
   color = "red", coord = "genome")
...
tracks(p1, p2, p3, p4,
   heights = c(2, 4, 1, 1))
```



- Layout genome linearly, stack associated data plots, connections

Examples



```
library(gridExtra)
grid.arrange(square,
   gg, ncol = 2,
   widths = c(4/5, 1/5))
```

Organize multiple circular layouts

Benefits

- Flexibility in drawing genomic data
- Aesthetics are changeable, color schemes for different purposes
- Plots defined in a way to compare and contrast
- Huge variety of displays is available in one location
- Builds from a good data model and tools available in bioC.

Future Work

- Clean up code, autoplot, consistency in usage, make circular layouts as elegant as Circos
- Ideally integrate new grammar components better with the ggplot2 code (not trivial)
- Build interactive graphics, using the qtbase, qtpaint primitives

Availability

- ggbio is on www.bioconductor.org
- Tengfei's ggbio web page has tutorials and gallery of examples: http://tengfei.github.com/ggbio
- Support by Genentech has been vital