A Brief Introduction to `ggplot2`

Bin Zhuo, PhD -
Biostatistician @ Celerion Inc.
-

November 15, 2016

software requirement

To make the result reproducible, the following packages are installed.

- `R` (>= 3.3.1)
- `Rstudio` (Version 1.0.44)
- `ggplot2` (>= 2.2.0)
- `rmarkdown` (>= 1.1)
- `revaeal_js` (>= 0.7)
- `knitr` (>= 1.14)

Instructions to reproduce this presentation slides

- unzip the package to your local drive
- open `ggplot2.Rmd` in `Rstudio`
- change the working directory to the folder where `dat.csv` is located (see slide 10).
- click `Knit` button in the source panel of `Rstudio`.

Why `ggplot2`?

- many users (among top five most frequently downloaded R packages)
- consistent underlying `grammar of graphics`
- plot specification at a high level of abstraction
- very flexible
- theme system for polishing plot appearance
- mature and complete graphics system

Materials and setup

Laptop users: You should have R installed, if not:

- Open a web browser and go to http://cran.r-project.org/ and download and install it
- Also helpful to install RStudio (download from http://rstudio.com/) [Strongly recommand!]
- In **R**, type `install.packages("ggplot2")` to install the `ggplot2` package.

What is the grammar of graphics

The basic idea: independently specify plot building blocks and combine them to create just about any kind of graphical display you want. Building blocks of a graph include:

- Data
- Geometric object
- Aesthetic mapping
- Theme
- Faceting
- Scale
- Annotation
- Coordinate system
- ...

1. Data —RNA-Sequencing gene expression

Key feature: high dimensional but small sample size

Gene	S1	S2	S3	S4	S5	S6
AT1G01010	136	140	128	110	83	119
AT1G01020	201	281	190	209	156	222
AT1G01030	2	4	8	2	4	6
AT1G01040	941	1291	849	1025	774	1084
AT1G01046	19	24	17	22	9	20
AT1G01050	1147	1268	1038	1091	833	1108
...	...	...	...	...	...	...

Arabidopsis thaliana

- model plant used for studying plant biology
- first plant to be completely sequenced (2000)
- small genome size, only 119 Mb (Human has > 3,000 Mb per haploid genome)

Data processing and summary

- Raw data downloaded from National Center for Biotechnology Information (NCBI) .
- Each sample may contain a text file of more than 10GB.
- May take several hours to process one sample on Linux computing cluster.
- Assembled my own pipeline to process the data.
- Over 2TB data are processed to get the counts from 211 samples.

Group	experiment	treatment	sample	gene
seedling	9	27	60	33602
leaf	5	28	60	33602
multi-tissue	10	39	91	33602

1. Data

```
# randomly sample 5 genes
# change this path to where dat.csv is located in your computer.
setwd("C:/Users/zhuob01/Desktop/ASA_Nebraska")
dat <- read.csv("dat.csv", header =T)
dat$Sample <- as.factor(dat$Sample)
dat$lab <- as.factor(dat$lab)
str(dat)
```

```
## 'data.frame':    455 obs. of  4 variables:
##  $ lab   : Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 2 2 2 2 ...
##  $ Sample: Factor w/ 91 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Gene  : Factor w/ 5 levels "AT1G64840","AT1G75420",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ CPM   : num  15.4 14.3 14.3 14.9 14.2 ...
```

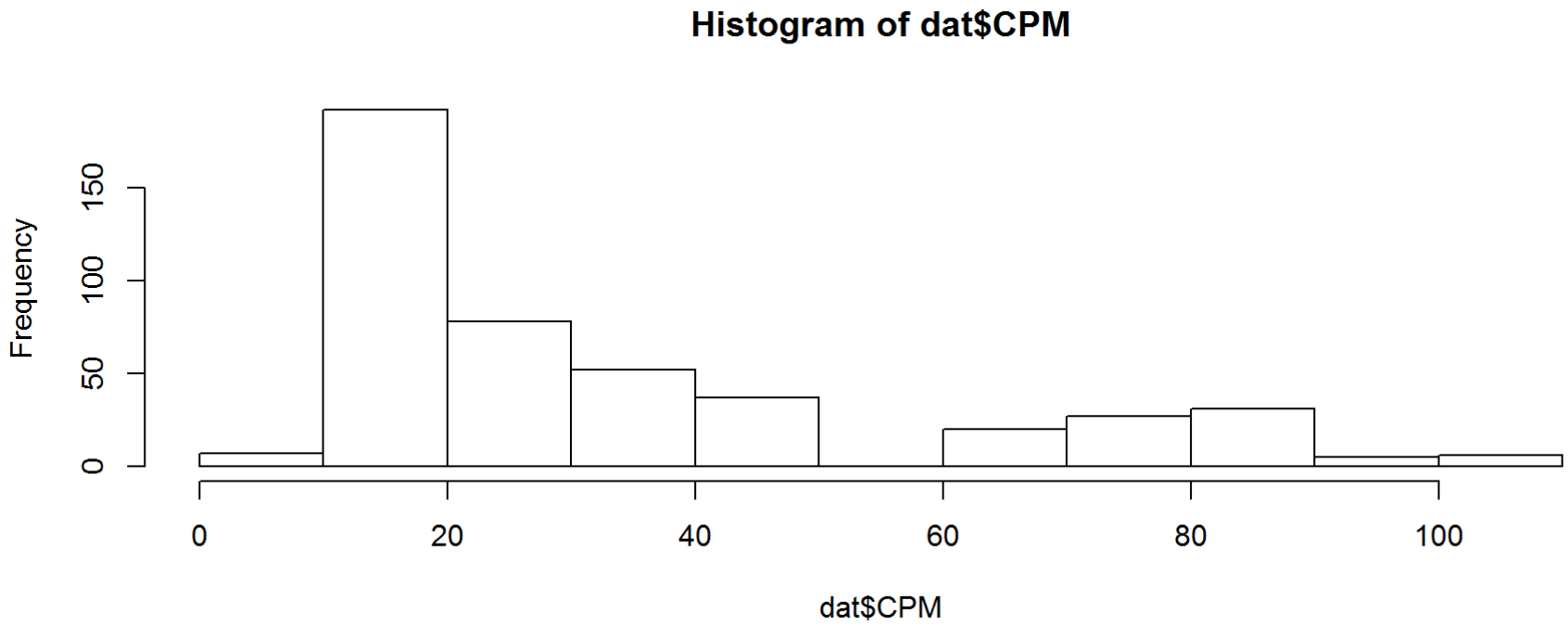
```
head(dat, 10)
```

```
##    lab Sample      Gene      CPM
## 1    1      1 AT1G64840 15.40205
## 2    1      2 AT1G64840 14.26605
## 3    1      3 AT1G64840 14.34562
## 4    1      4 AT1G64840 14.90372
## 5    1      5 AT1G64840 14.17669
## 6    1      6 AT1G64840 17.40853
## 7    2      7 AT1G64840 17.71515
## 8    2      8 AT1G64840 16.44038
## 9    2      9 AT1G64840 15.65335
## 10   2     10 AT1G64840 14.96202
```

Example – histogram

basic graphic histogram

```
hist(dat$CPM)
```

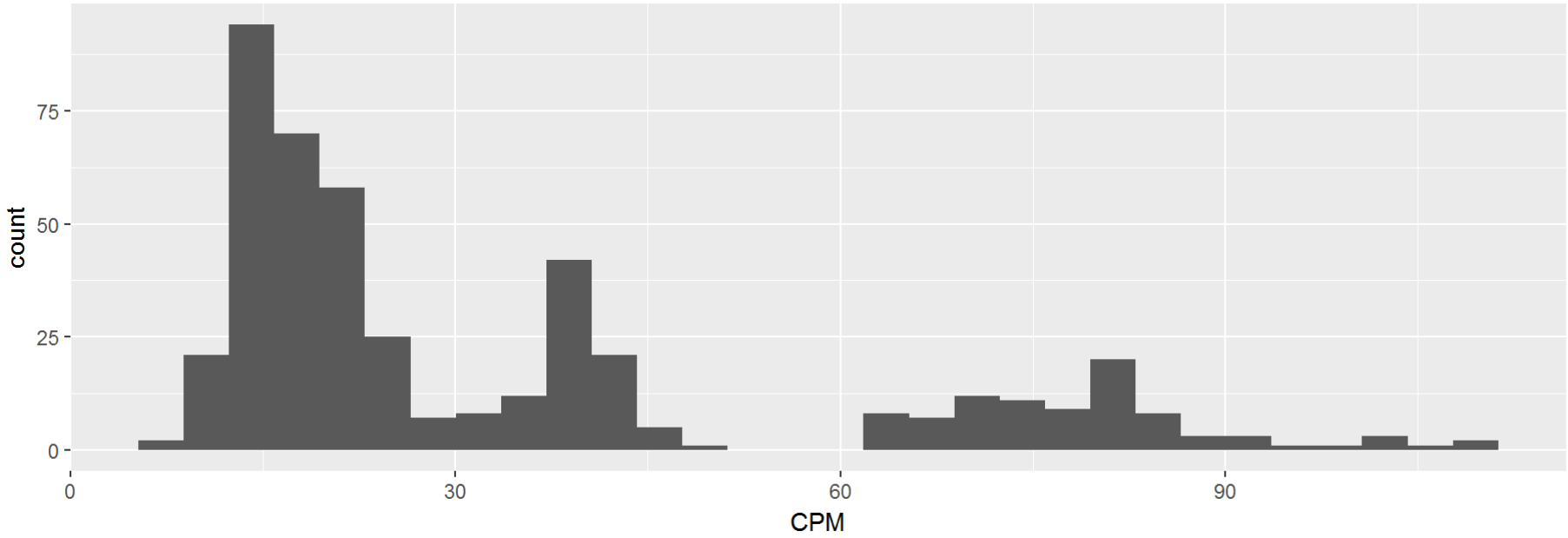


Example – histogram

`ggplot2` histogram

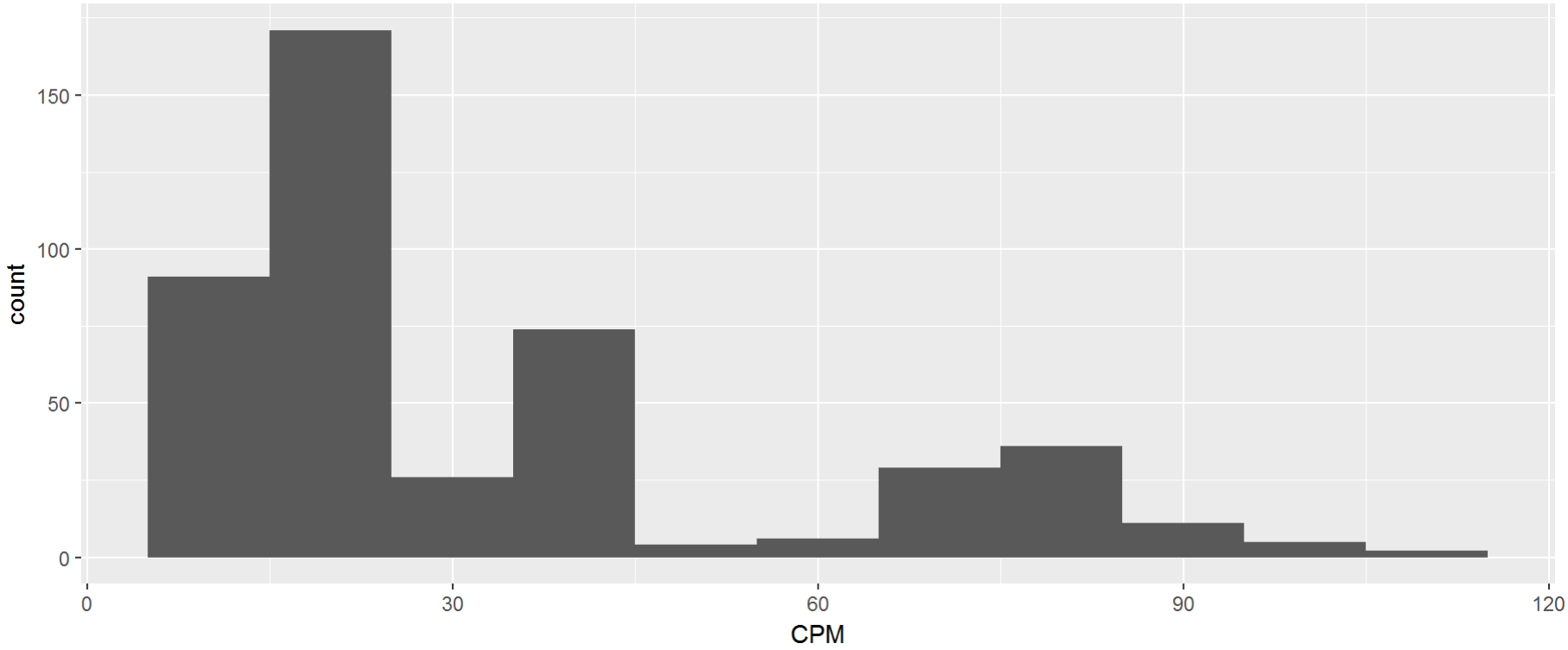
```
ggplot(dat, aes(x= CPM)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Change binwidth

```
ggplot(dat, aes(x= CPM)) + geom_histogram(binwidth = 10)
```



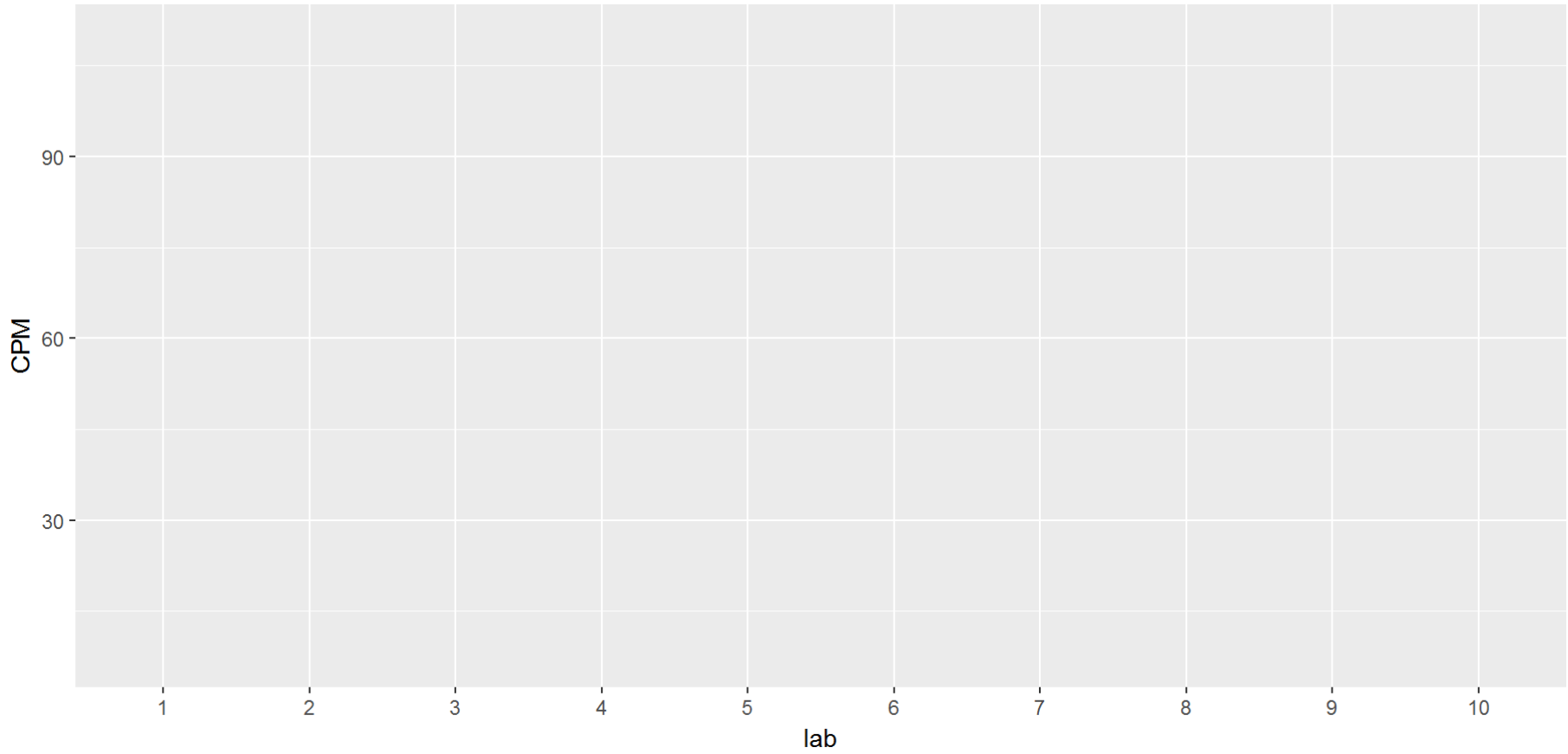
ggplot2 versus base

Compared to base graphics, `ggplot2`

- is more verbose for simple / canned graphics
- is less verbose for complex / custom graphics
- does not have methods (data should always be in a `data.frame`)
- uses a different system for adding plot elements

Basic Panel

```
p1 <- ggplot(dat, aes(x = lab, y = CPM))
p1
```

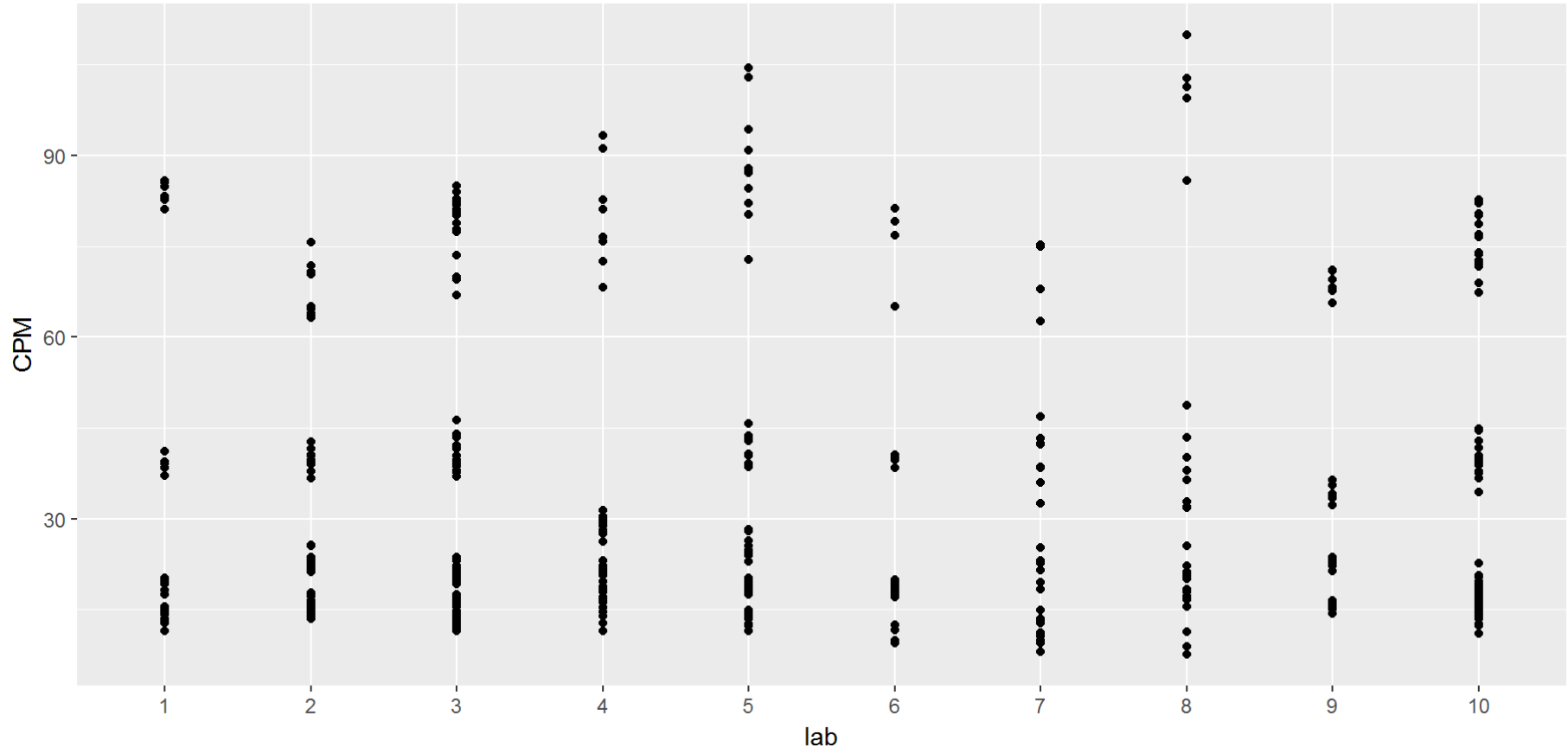


2. Geometric Objects

- points (`geom_point`)
- line (`geom_line`)
- histogram (`geom_histogram`)
- contour (`geom_contour`)
- density (`geom_density`)
- ...

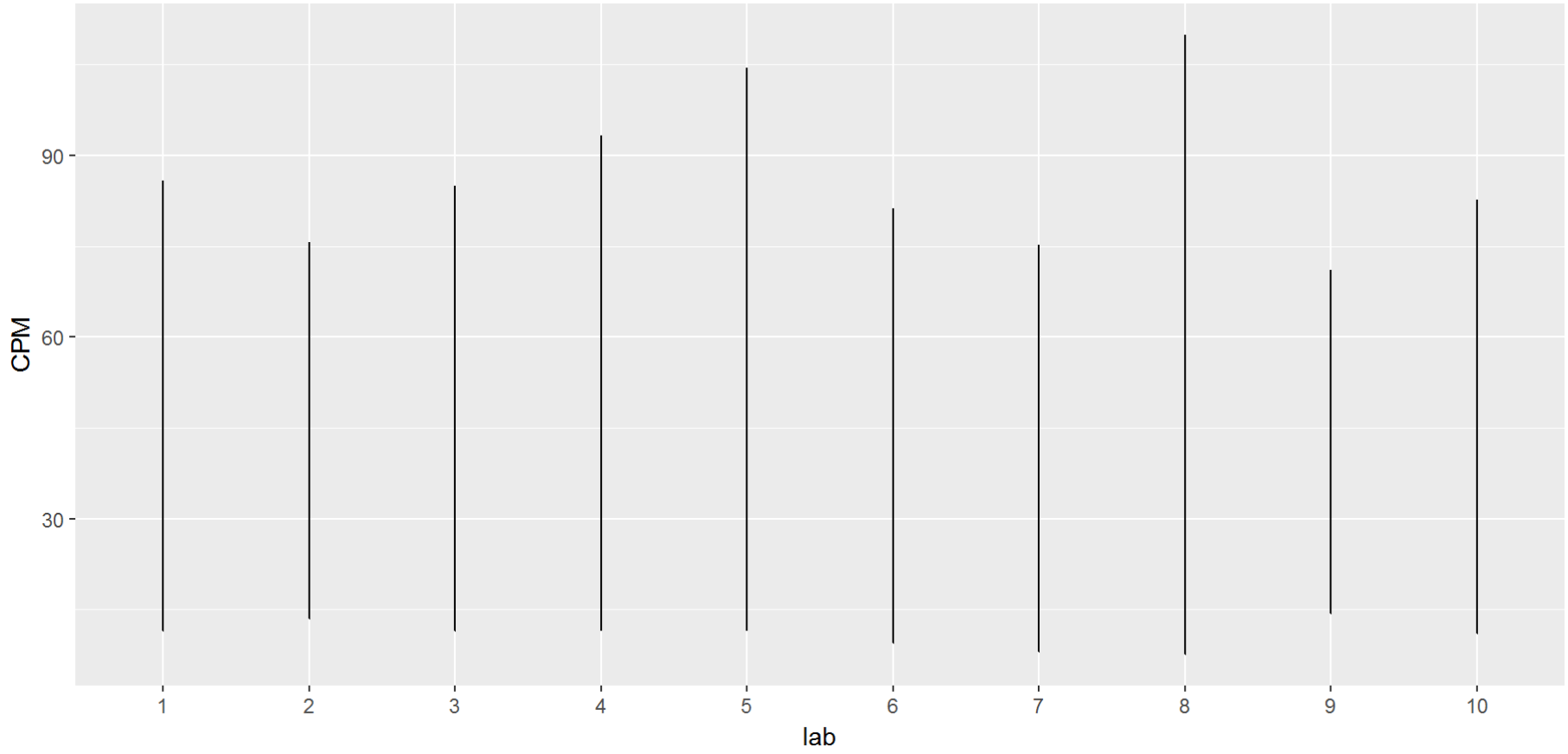
Assign a geometric object

```
p1 + geom_point()
```

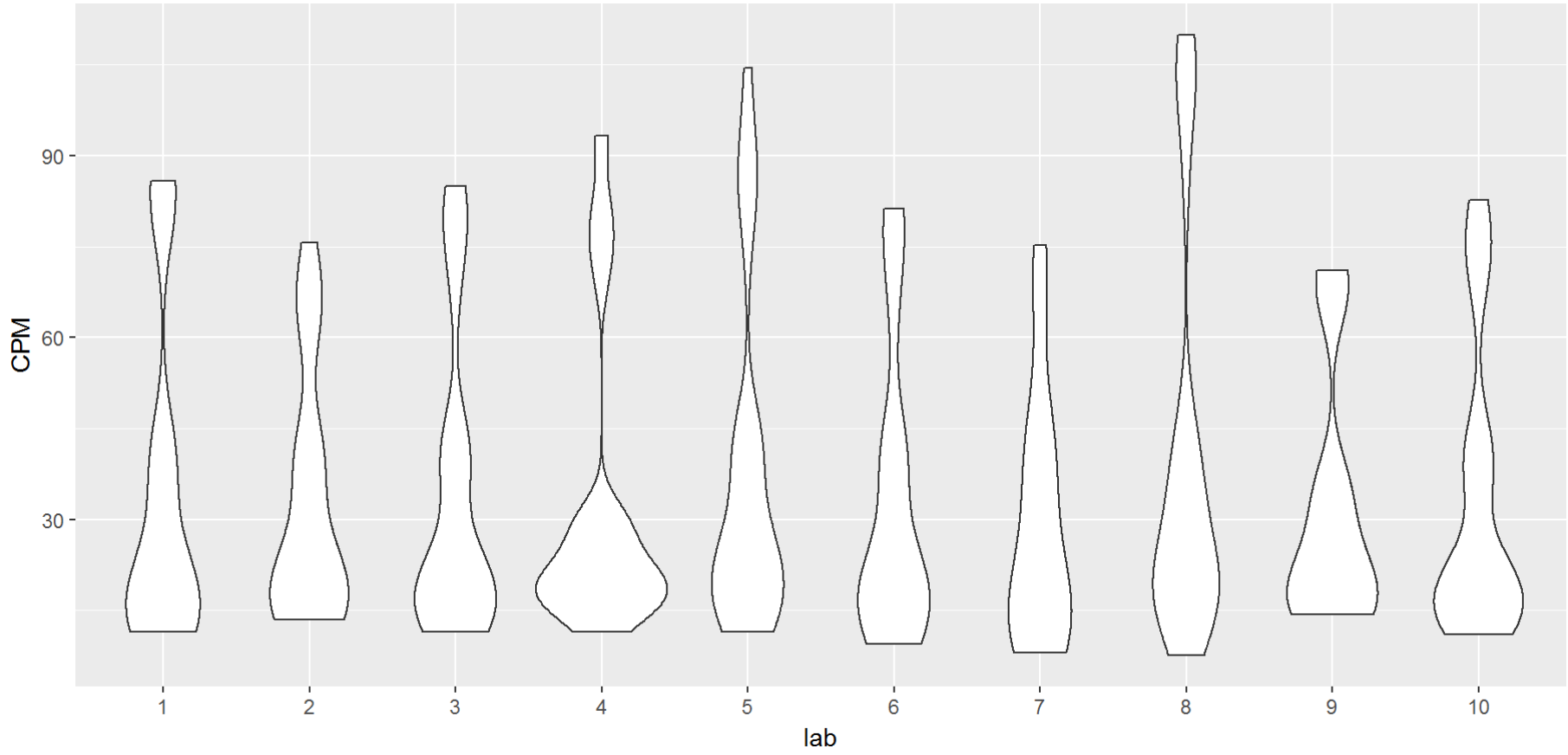


Change geometric object to lines

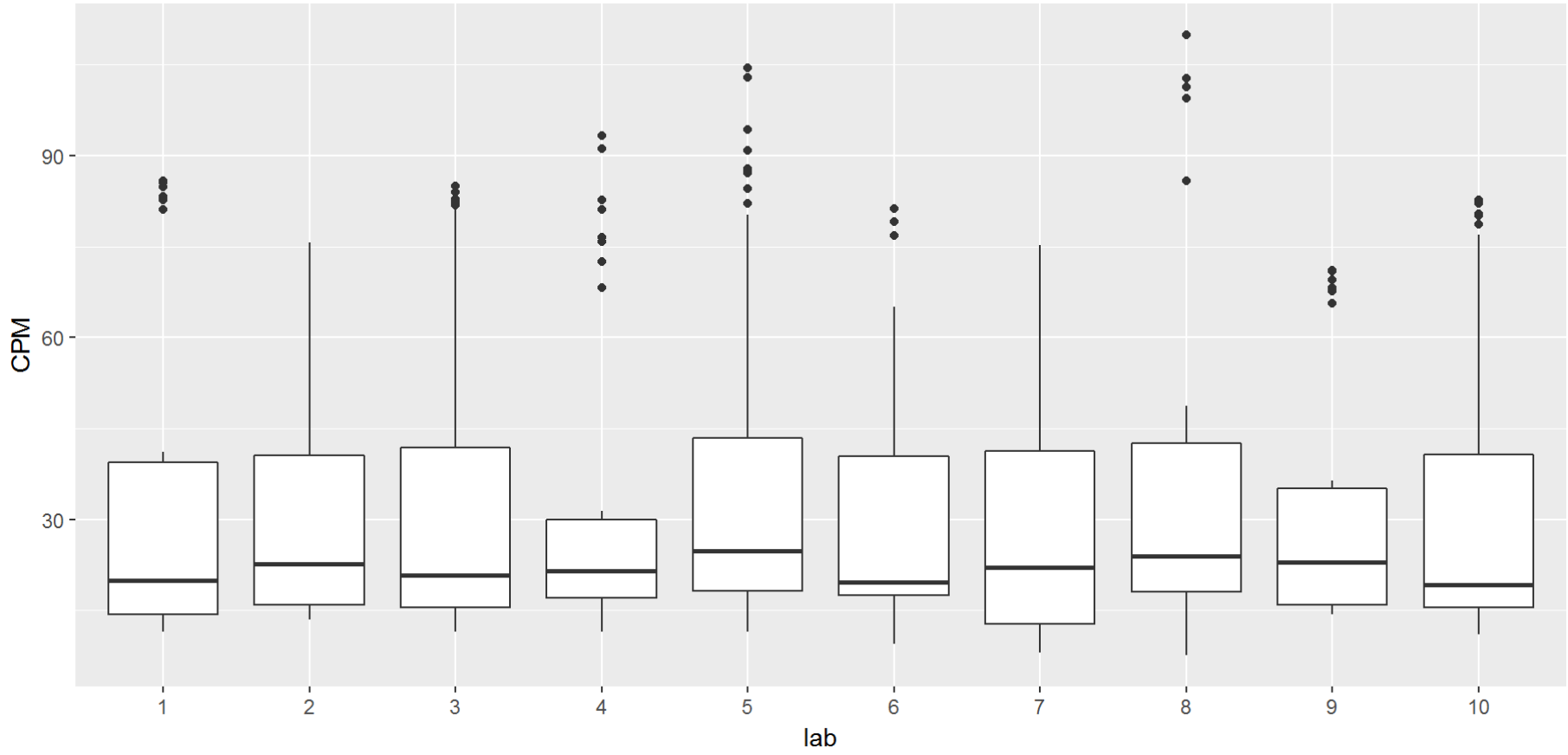
```
p1 + geom_line()
```



```
p1 + geom_violin()
```

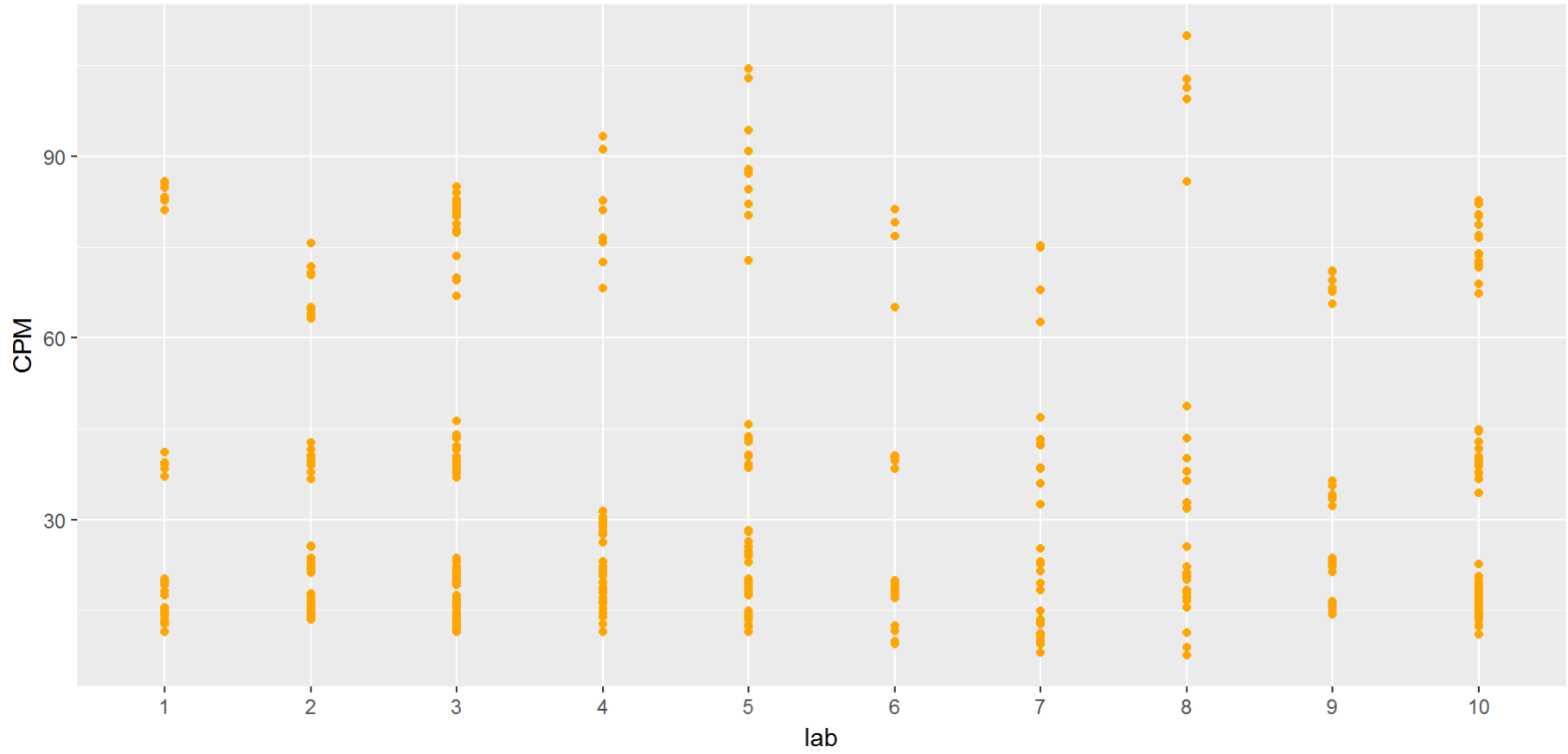


```
p1 + geom_boxplot()
```



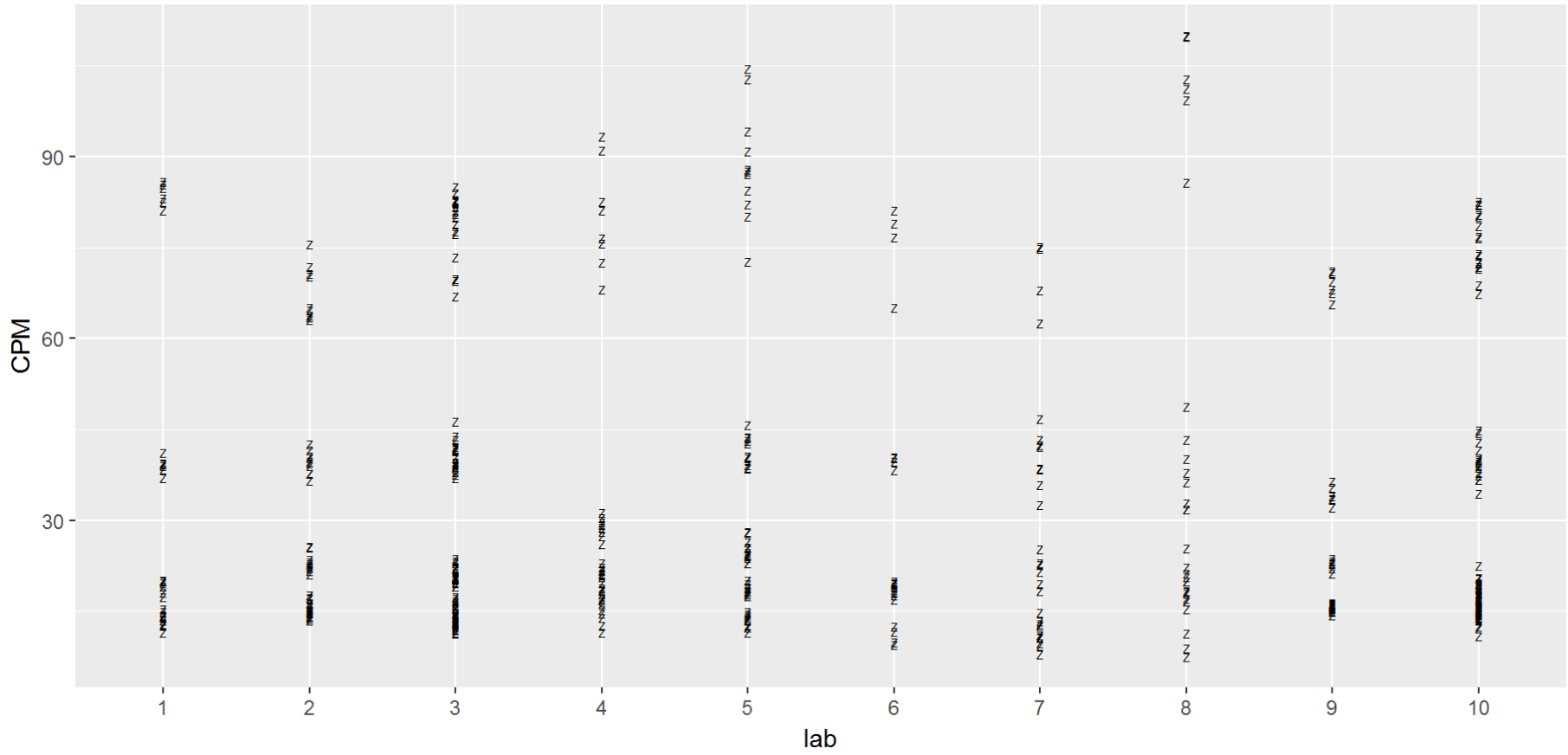
color

```
p1 + geom_point(color = "orange")
```



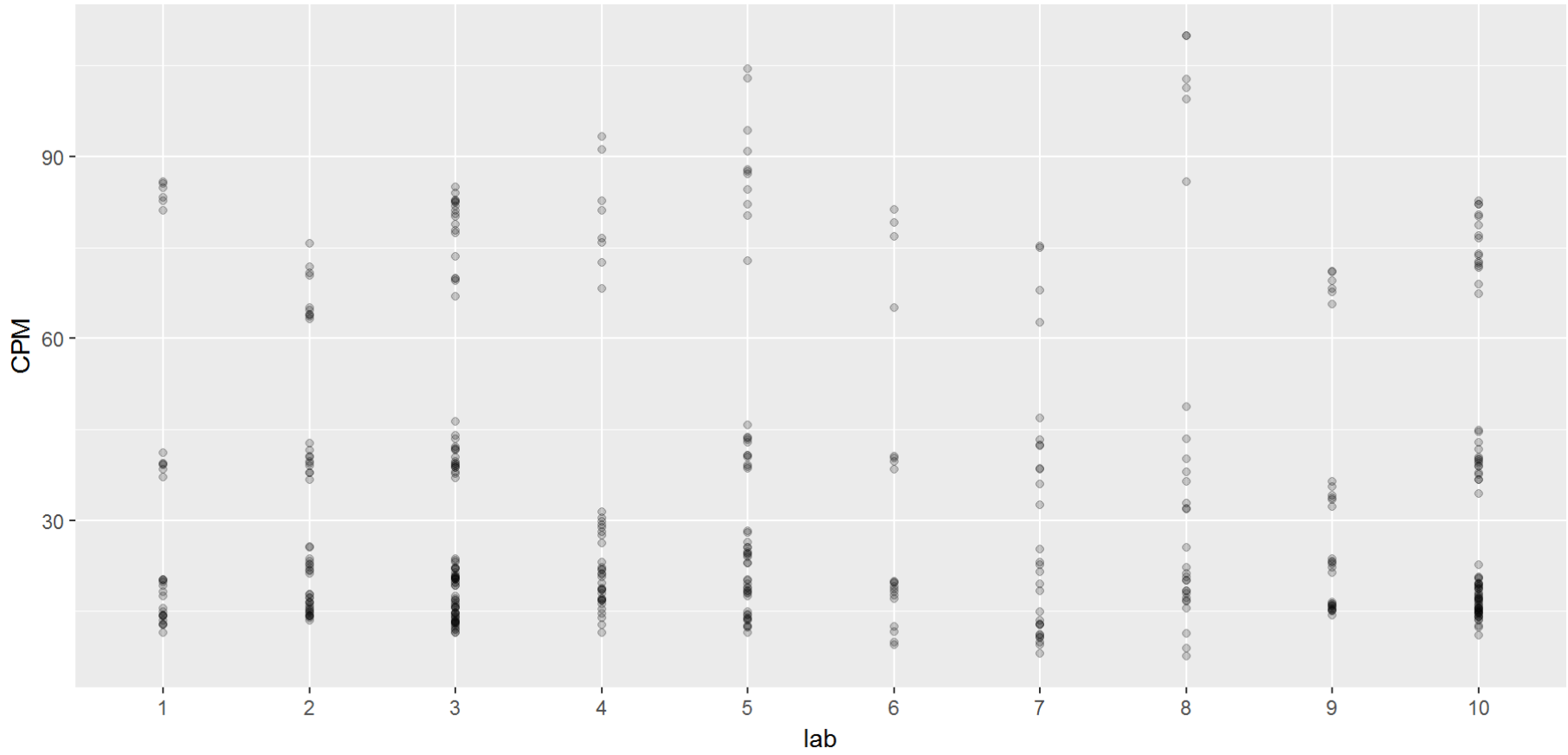
shape

```
p1 + geom_point(shape = "Z")
```



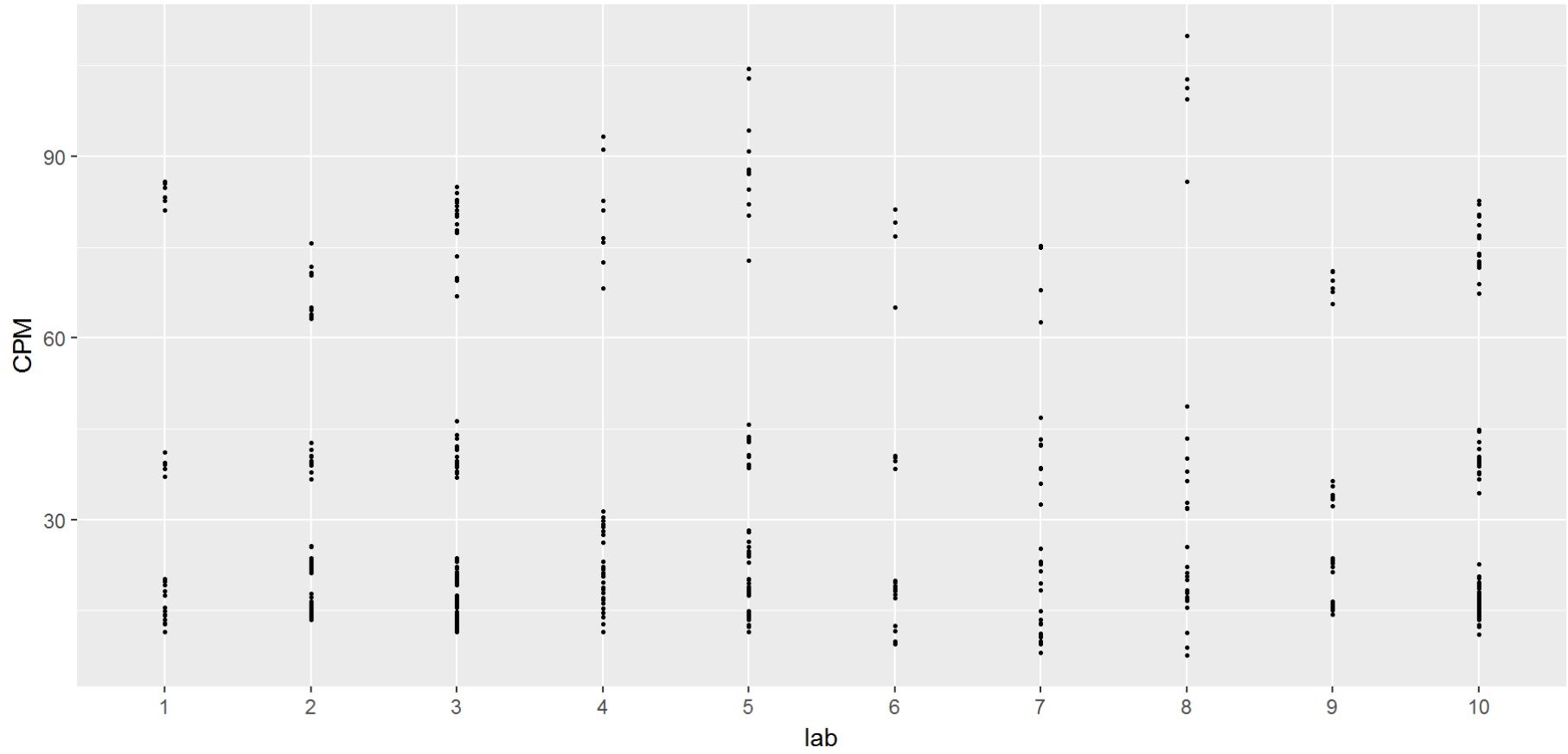
alpha

```
p1 + geom_point(alpha = .2)
```

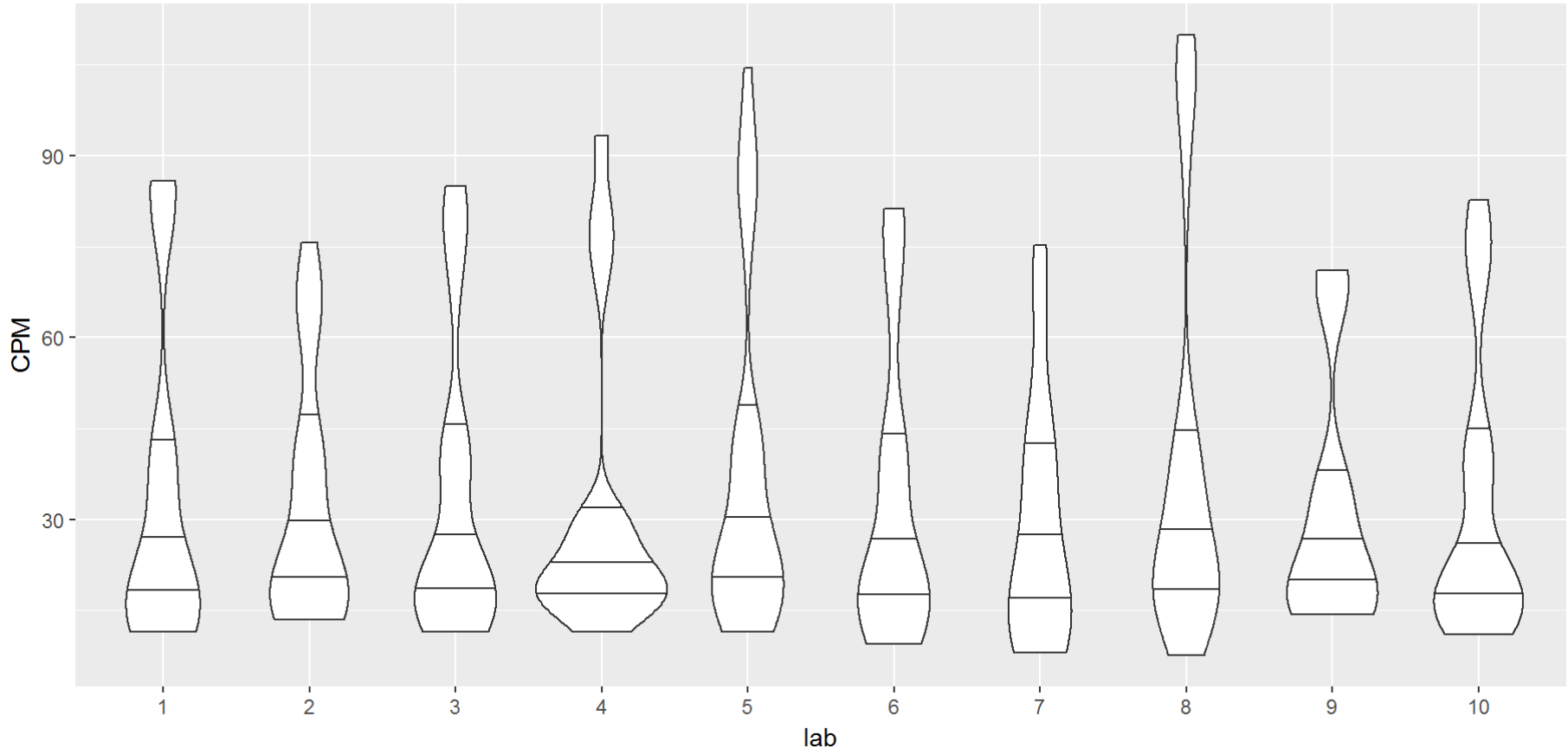


size

```
p1 + geom_point(size = 0.5)
```



```
p1 + geom_violin(draw_quantiles = c(0.25, 0.5, 0.75))
```

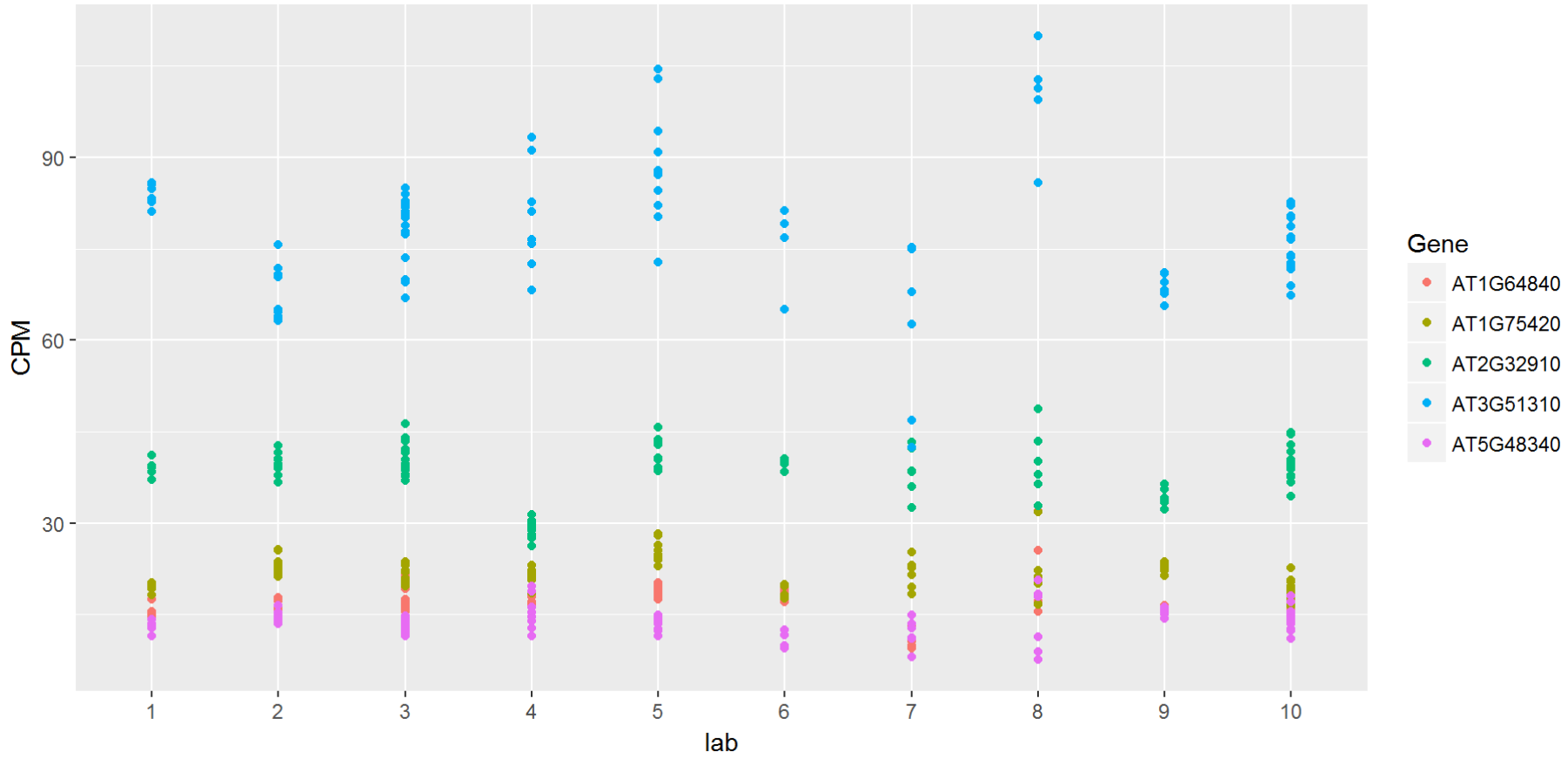


3. Aesthetic mapping

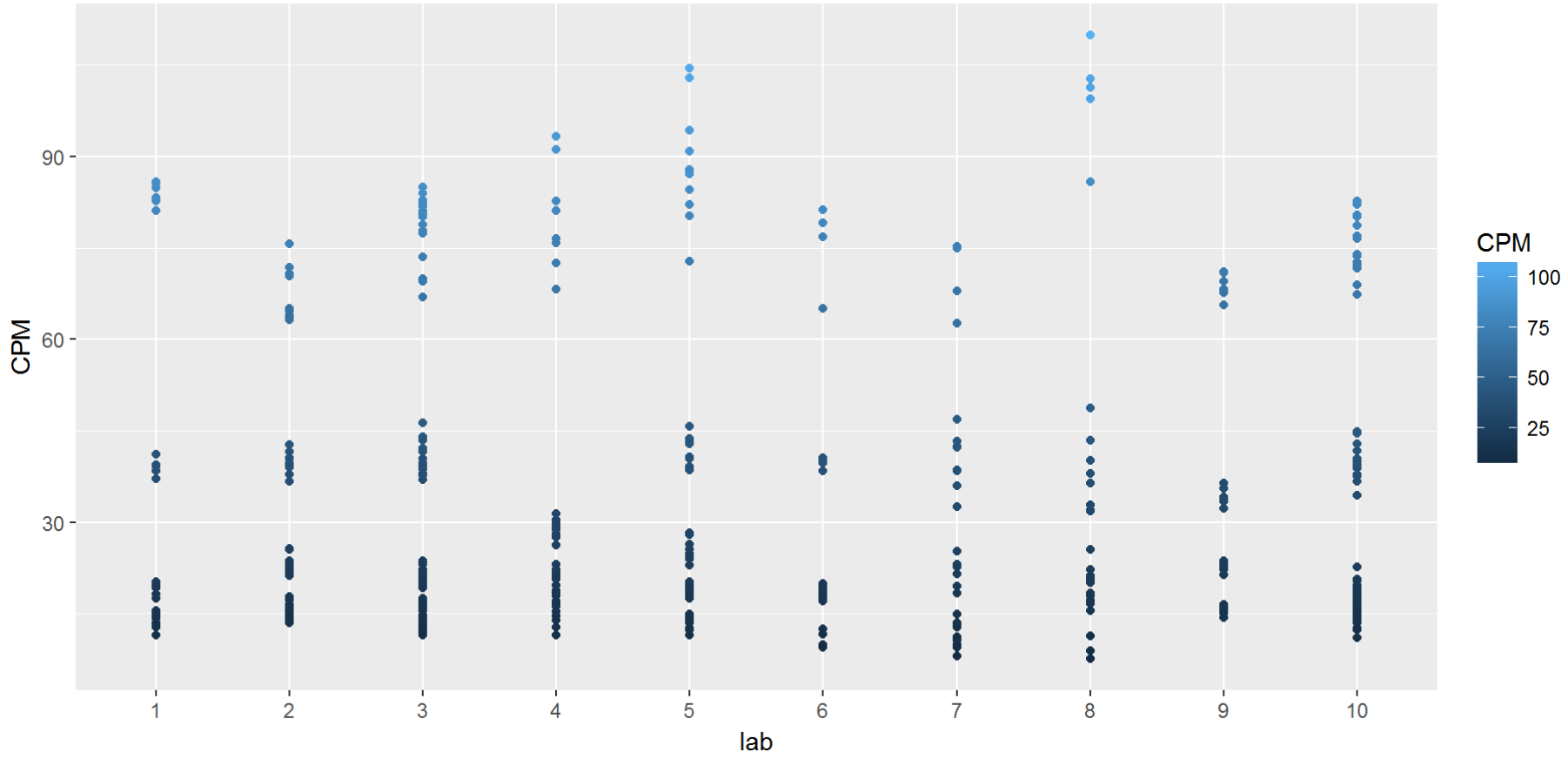
- `x` what should be on x-axis?
- `y` what should be on y-axis?
- `alpha`
- `color`
- `shape`
- `size`
- `stroke`
- ...

color

```
p1 + geom_point(aes(color = Gene))
```

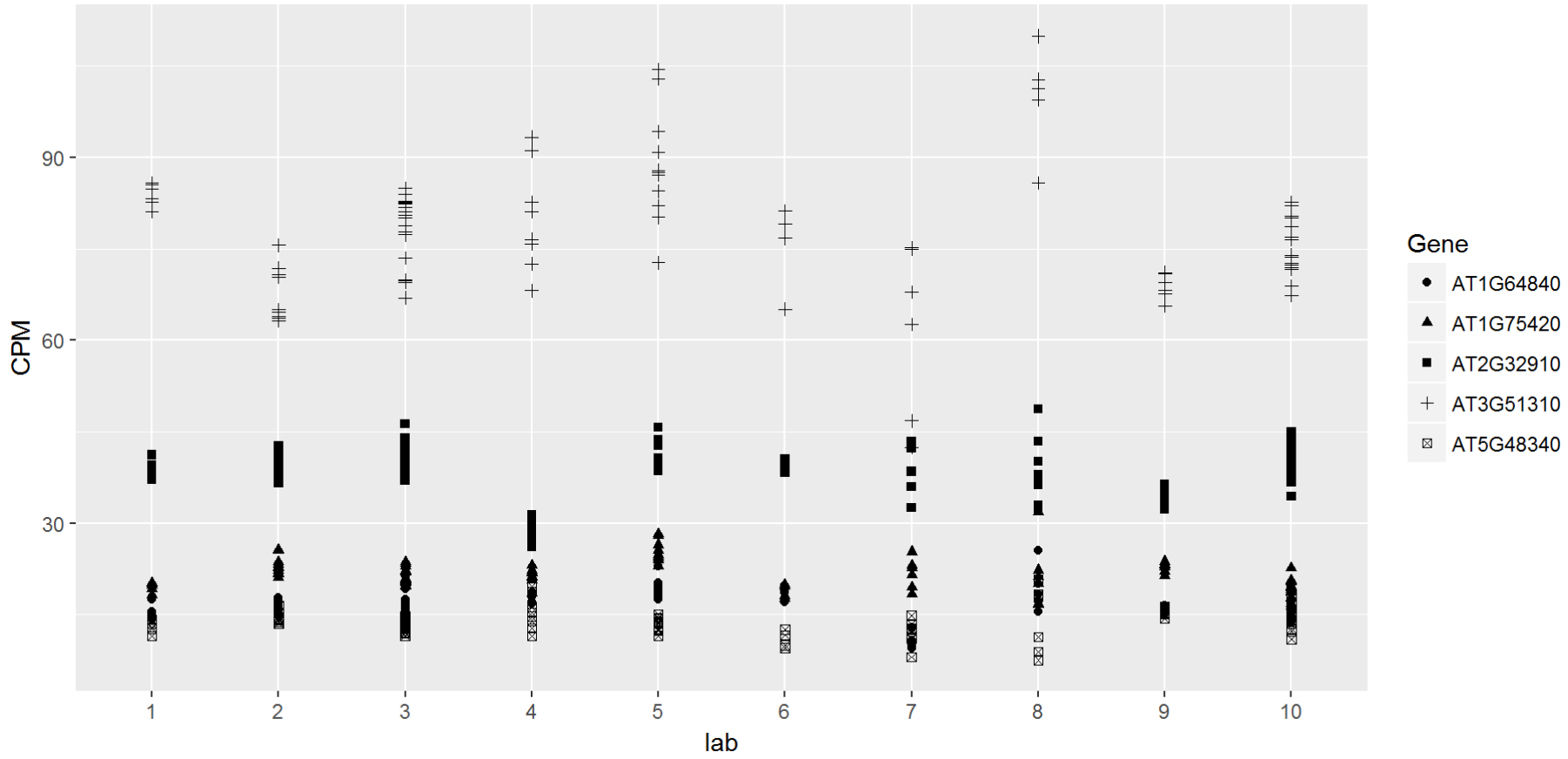


```
p1 + geom_point(aes(color = CPM))
```



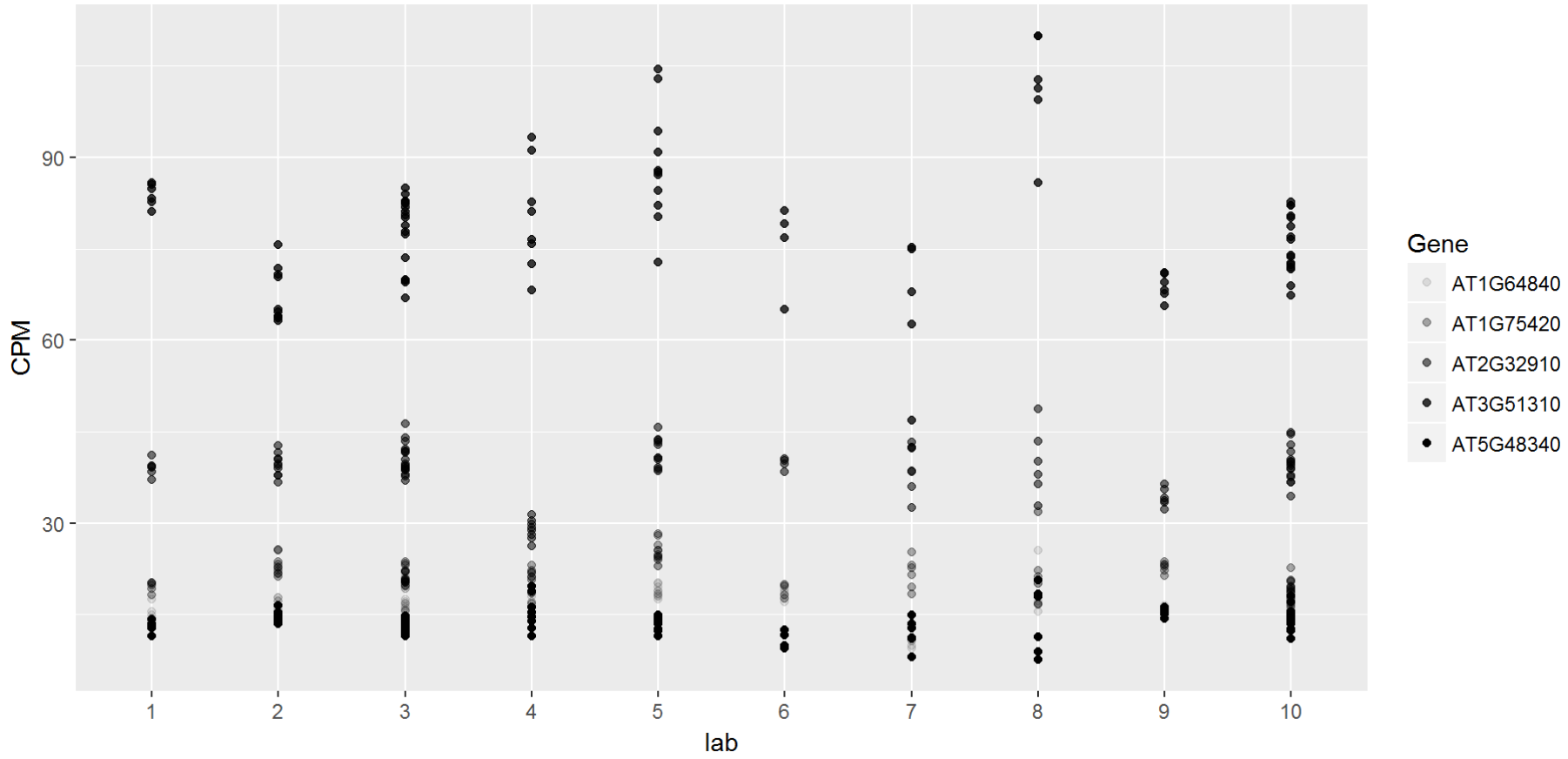
shape

```
p1 + geom_point(aes(shape = Gene))
```



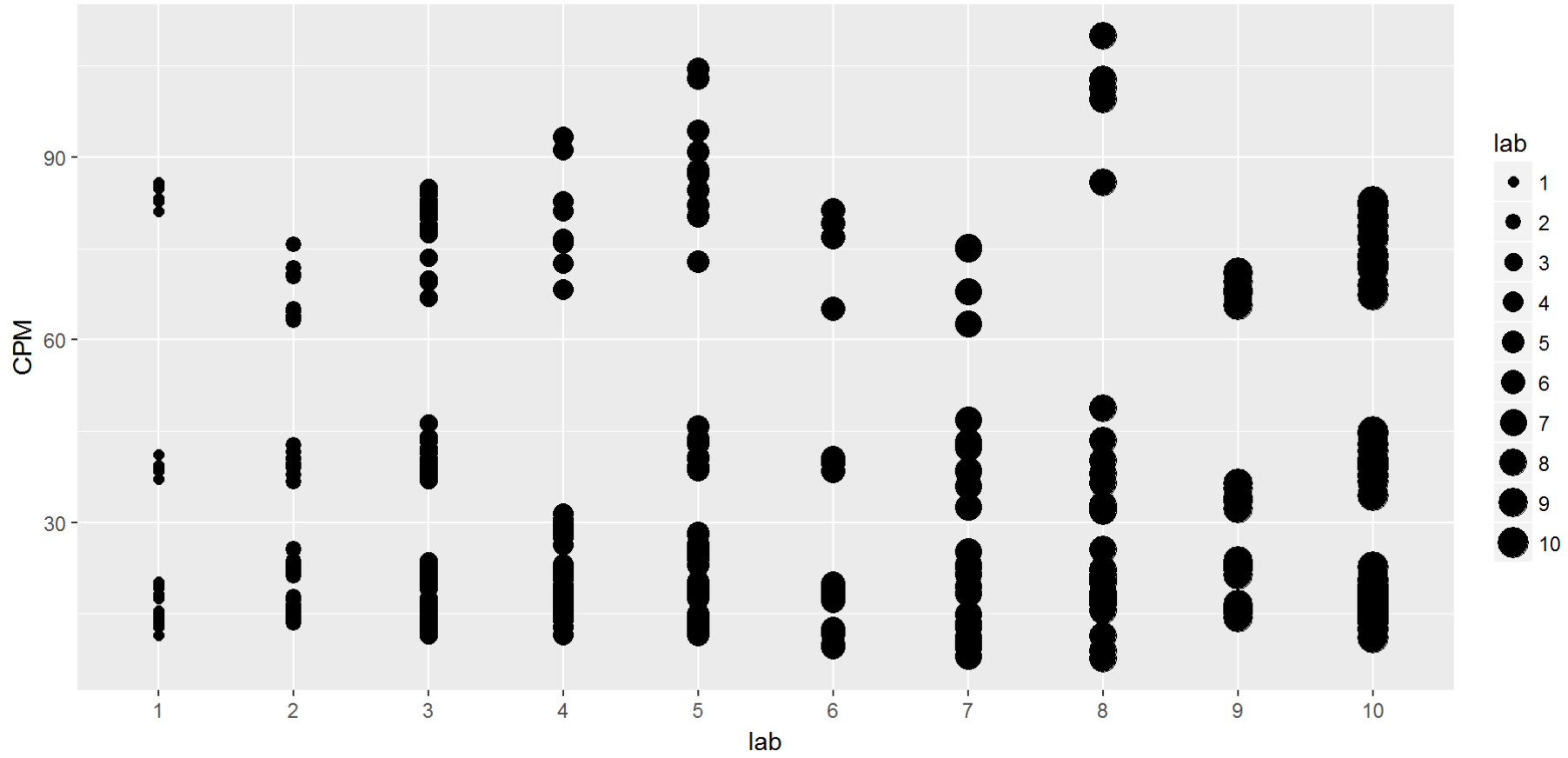
alpha

```
p1 + geom_point(aes(alpha = Gene))
```



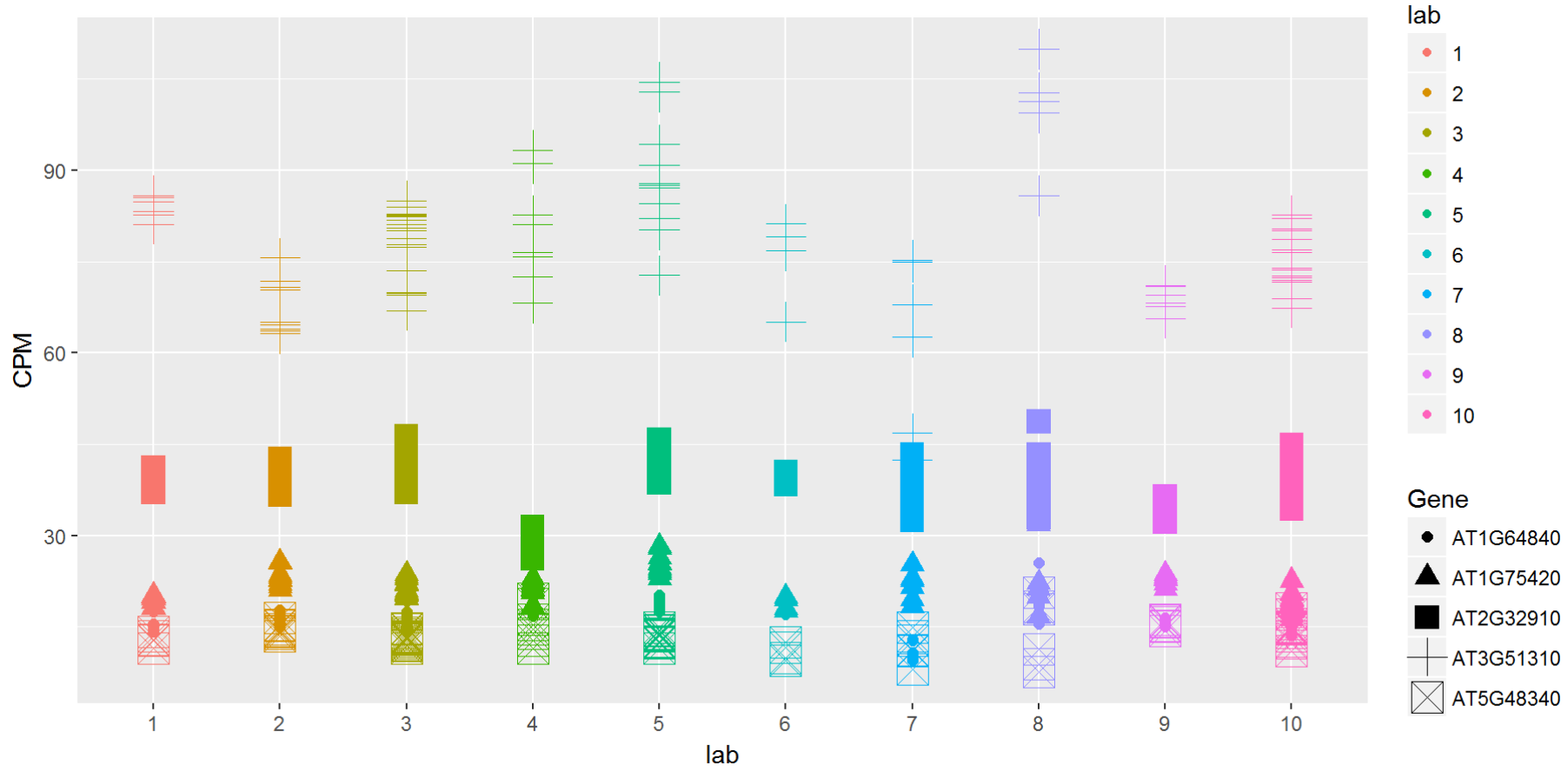
size

```
p1 + geom_point(aes(size = lab))
```

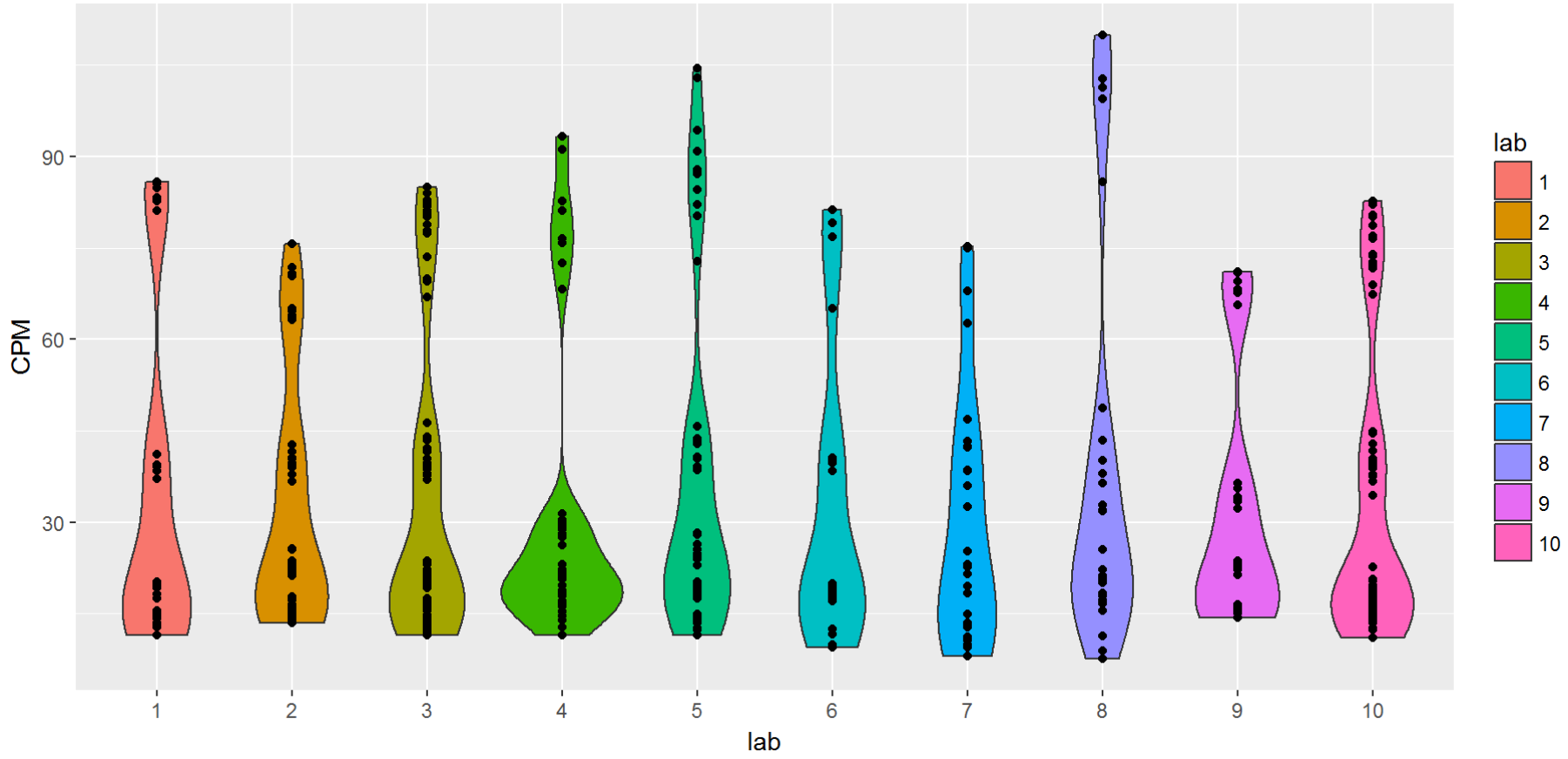


Combination

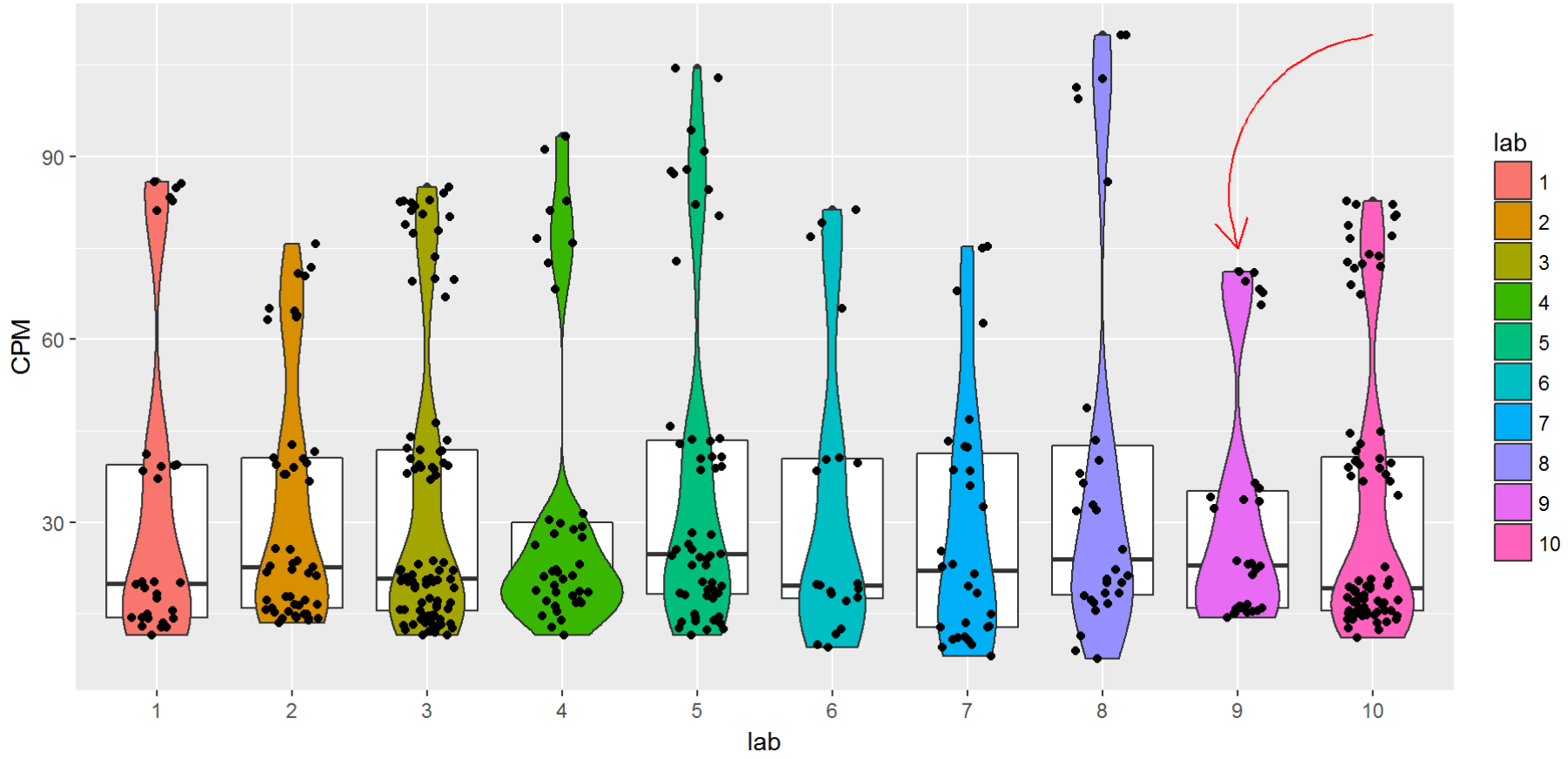
```
p1 + geom_point(aes(size = Gene, color = lab, shape = Gene))
```



```
p1 + geom_violin(aes(fill = lab)) + geom_jitter(width= 0)
```

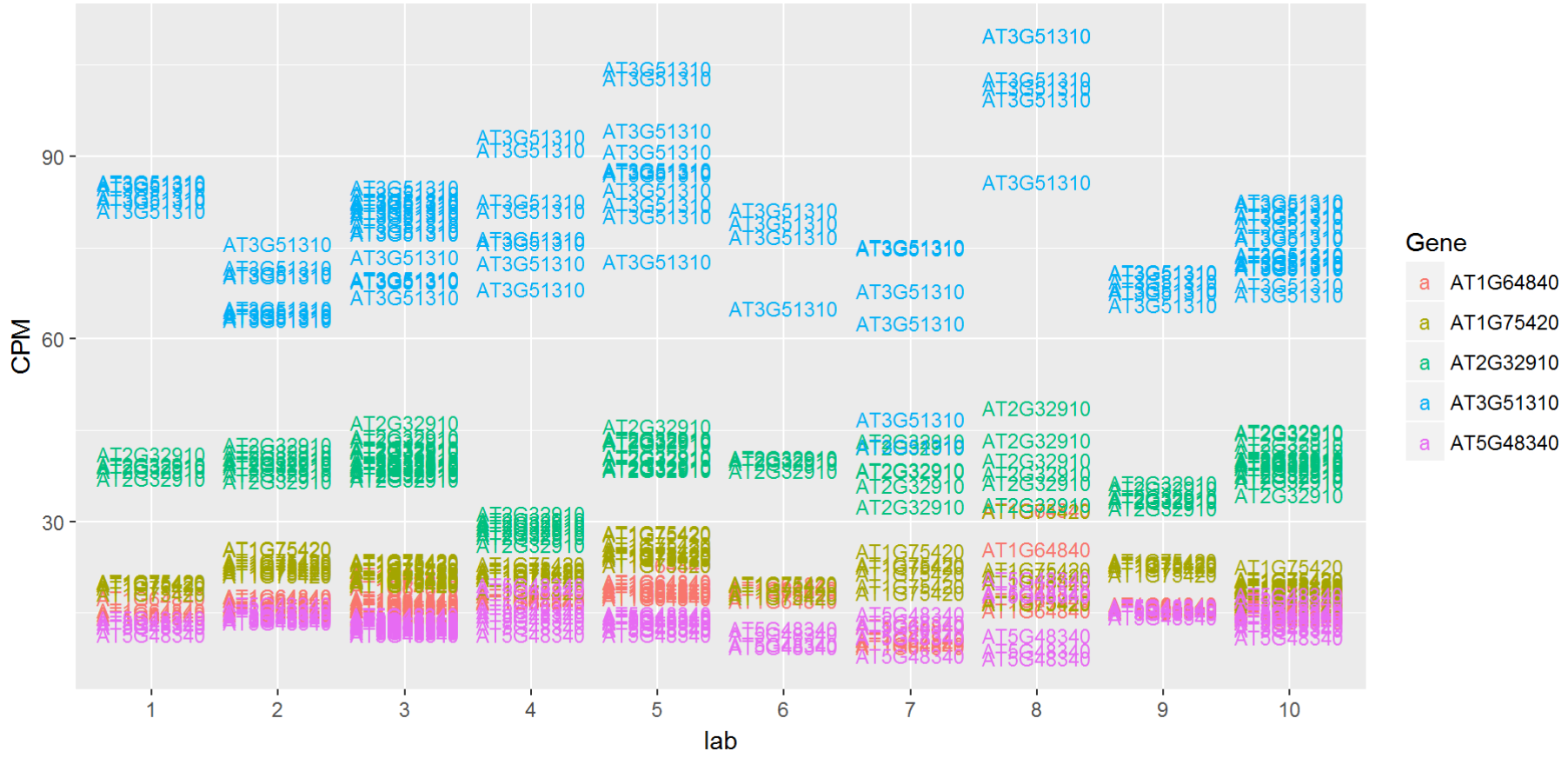


```
p1 + geom_boxplot()+ geom_violin(aes(fill = lab)) +
  geom_jitter(width = 0.2) +
  geom_curve(x =10, y = 110, xend = 9, yend= 75,
             arrow = arrow(length= unit(0.5, "cm")), color = "red")
```

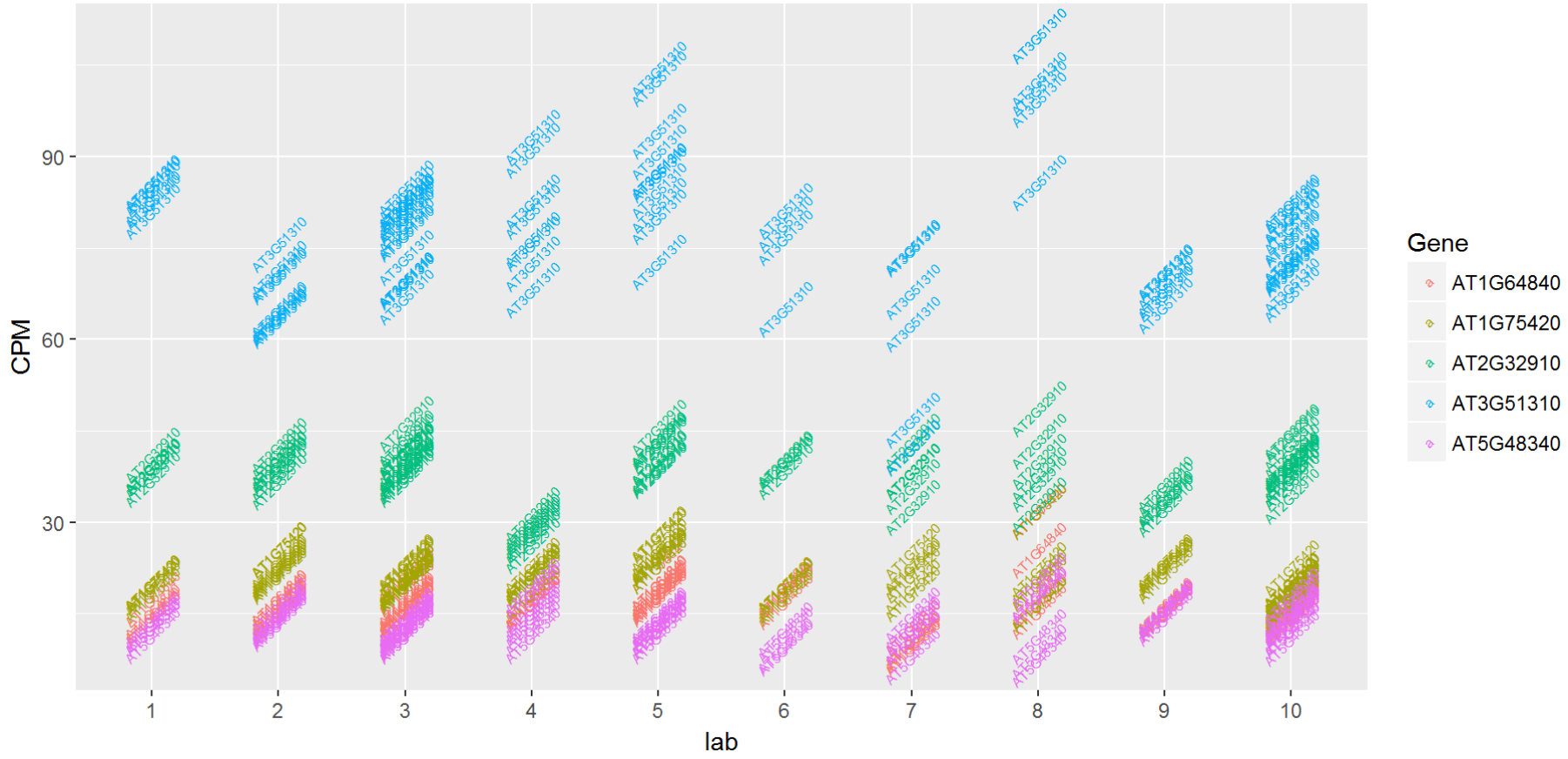


Use text!

```
p1 + geom_text(aes(label=Gene, color = Gene), size = 3)
```



```
p1 + geom_text(aes(label=Gene, color = Gene), size = 2, angle = 45)
```

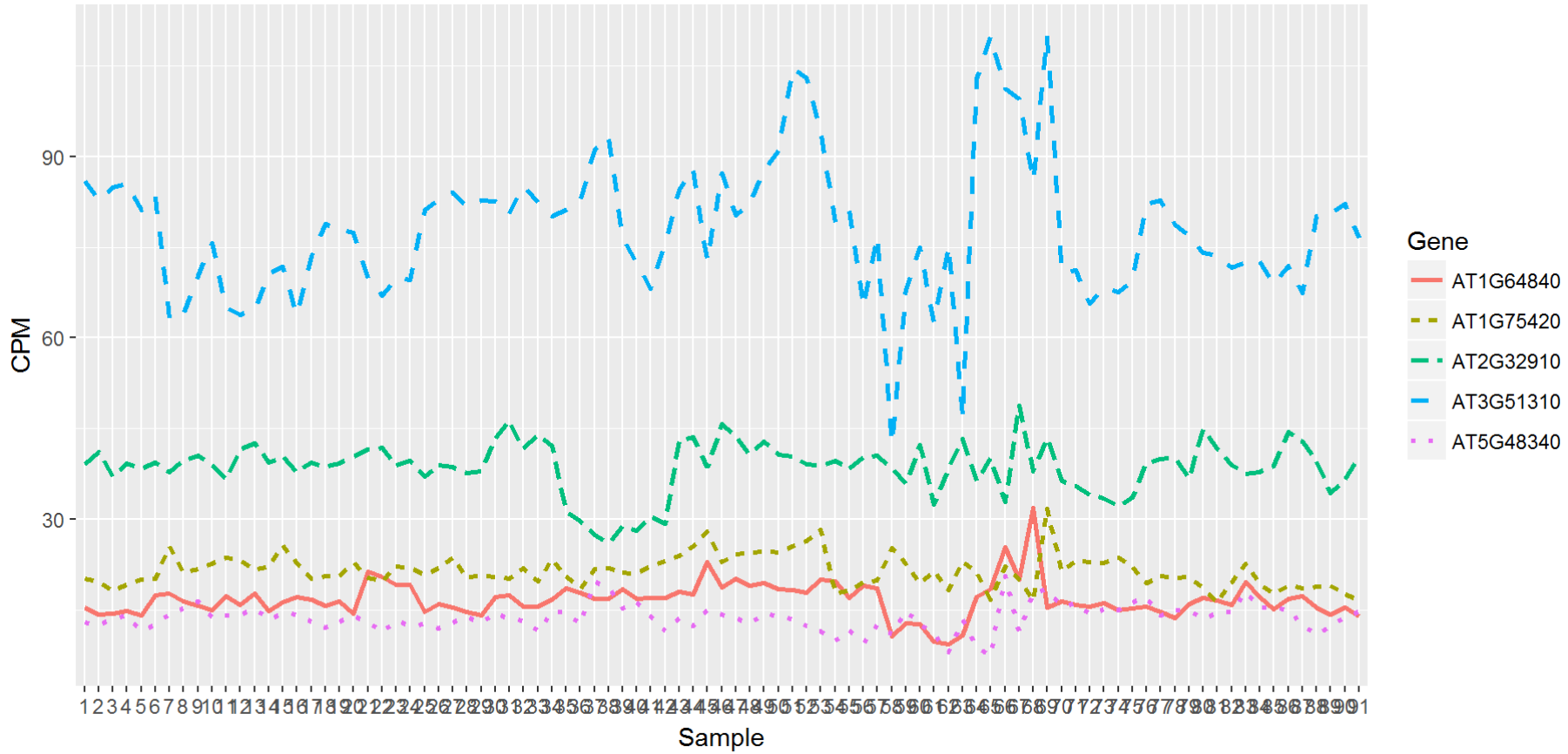


4. Themes

Themes control non-data components of the plot.

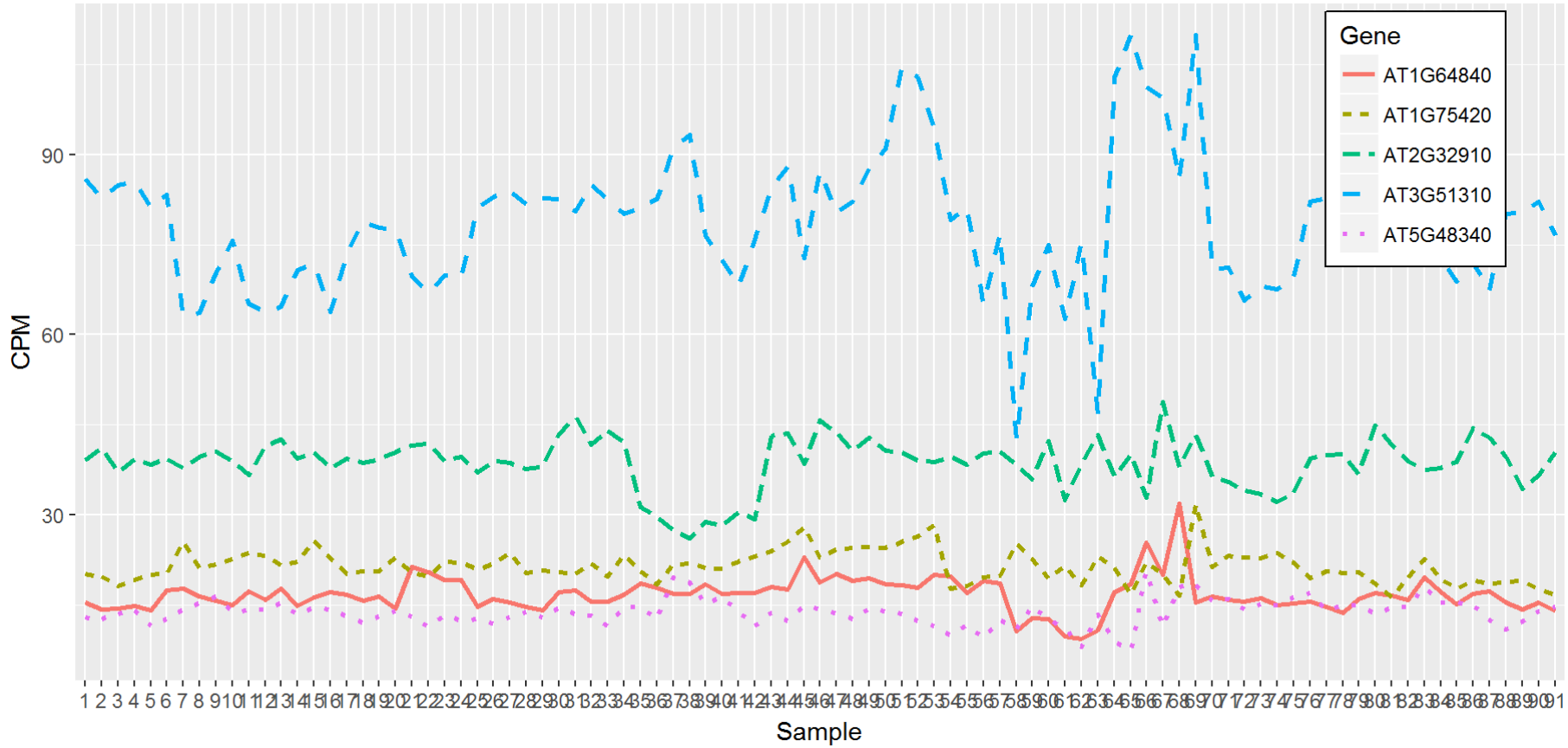
- `element_line` controls all line elements
- `element_rect` controls all rectangular elements
- `element_text` controls all text elements
- ...

```
s1 <- ggplot(dat, aes(x = Sample, y = CPM)) +
  geom_line(aes(linetype = Gene, color = Gene, group = Gene), size = 1)
print(s1)
```



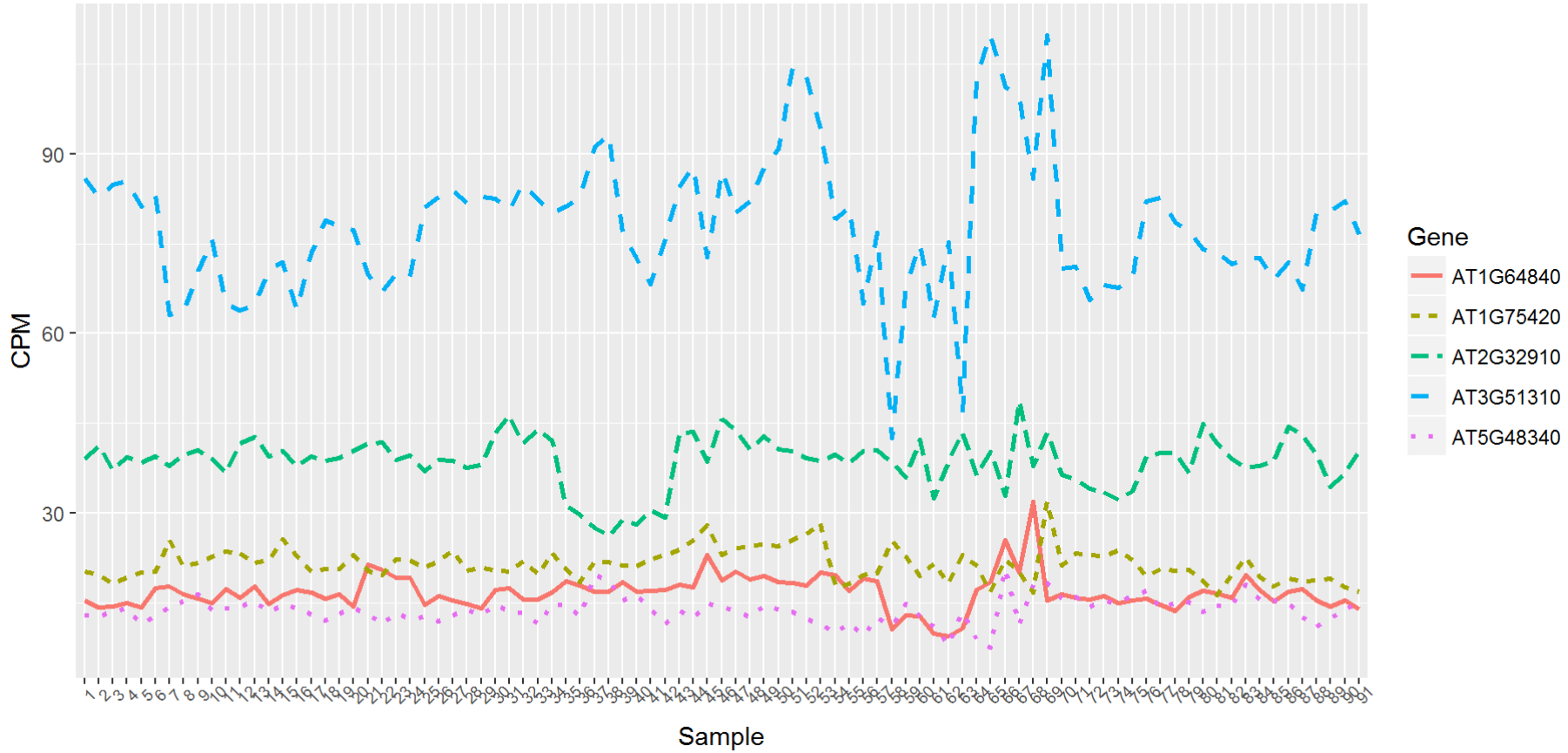
Themes - legend

```
s1 + theme(legend.position = c(.9, .8), legend.background = element_rect(colour = "black"))
```



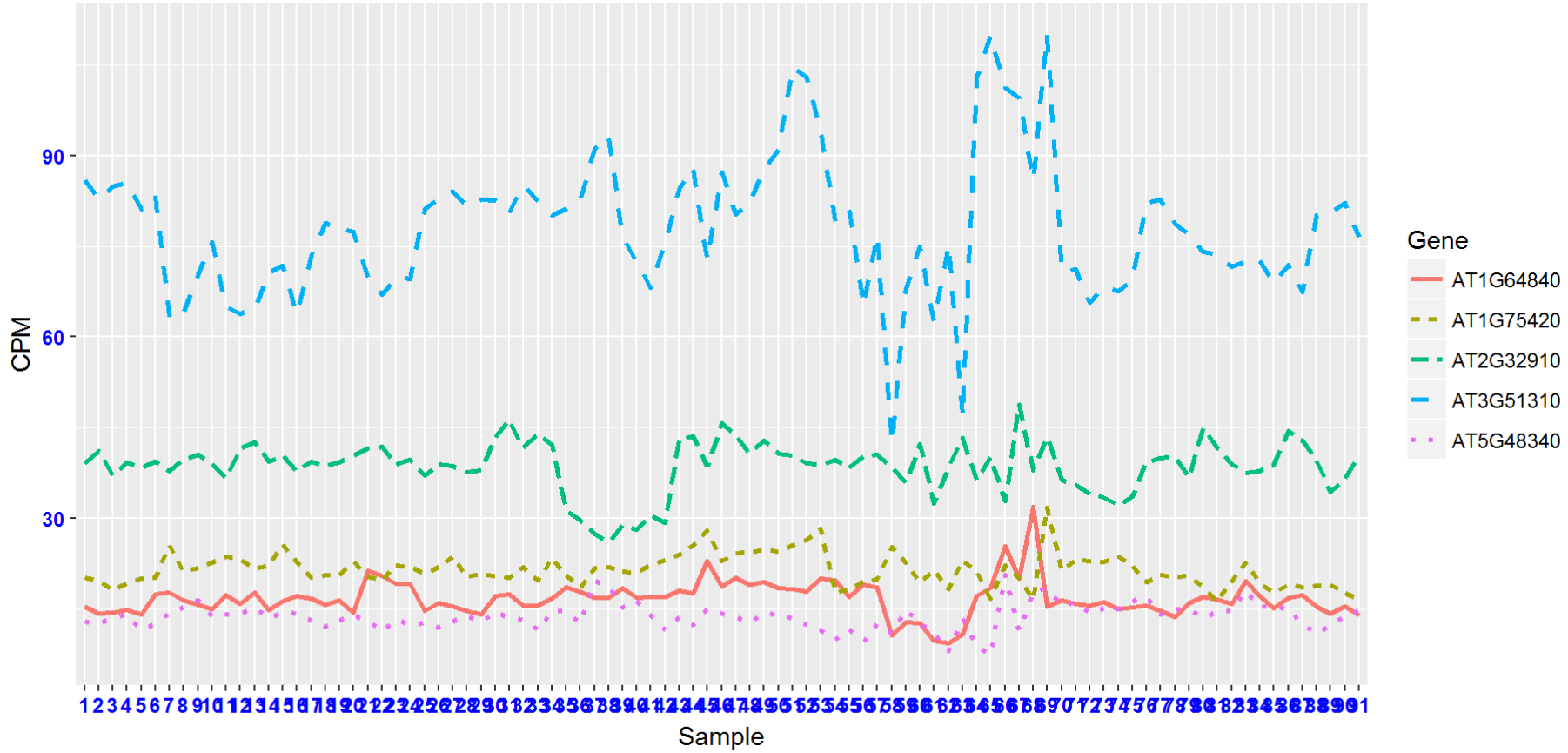
Themes - text size

```
s1 + theme(axis.text.x = element_text(size = 7, angle = 45))
```



Themes - text font

```
s1 + theme(axis.text = element_text(color = "blue", face = "bold"))
```



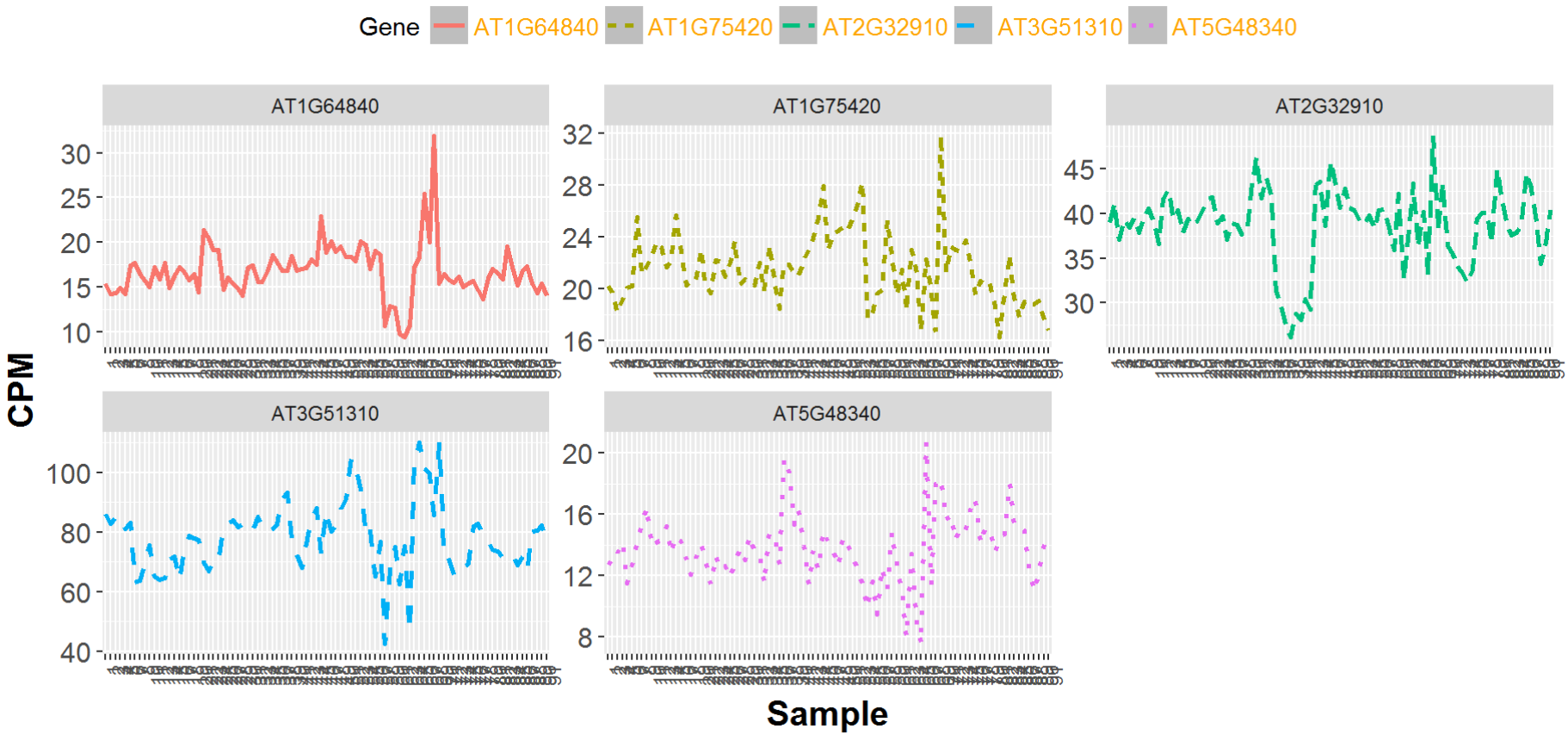
```
s2 <- s1 + theme(legend.position = "top", legend.key = element_rect(fill = "grey"),
                 legend.text = element_text(size = 10, color = "orange"),
                 axis.text.y = element_text(size = 12),
                 axis.text.x = element_text(size = 8, angle = 90, hjust = 1),
                 axis.title = element_text(size = 15, face = "bold"))
print(s2)
```

5. Faceting

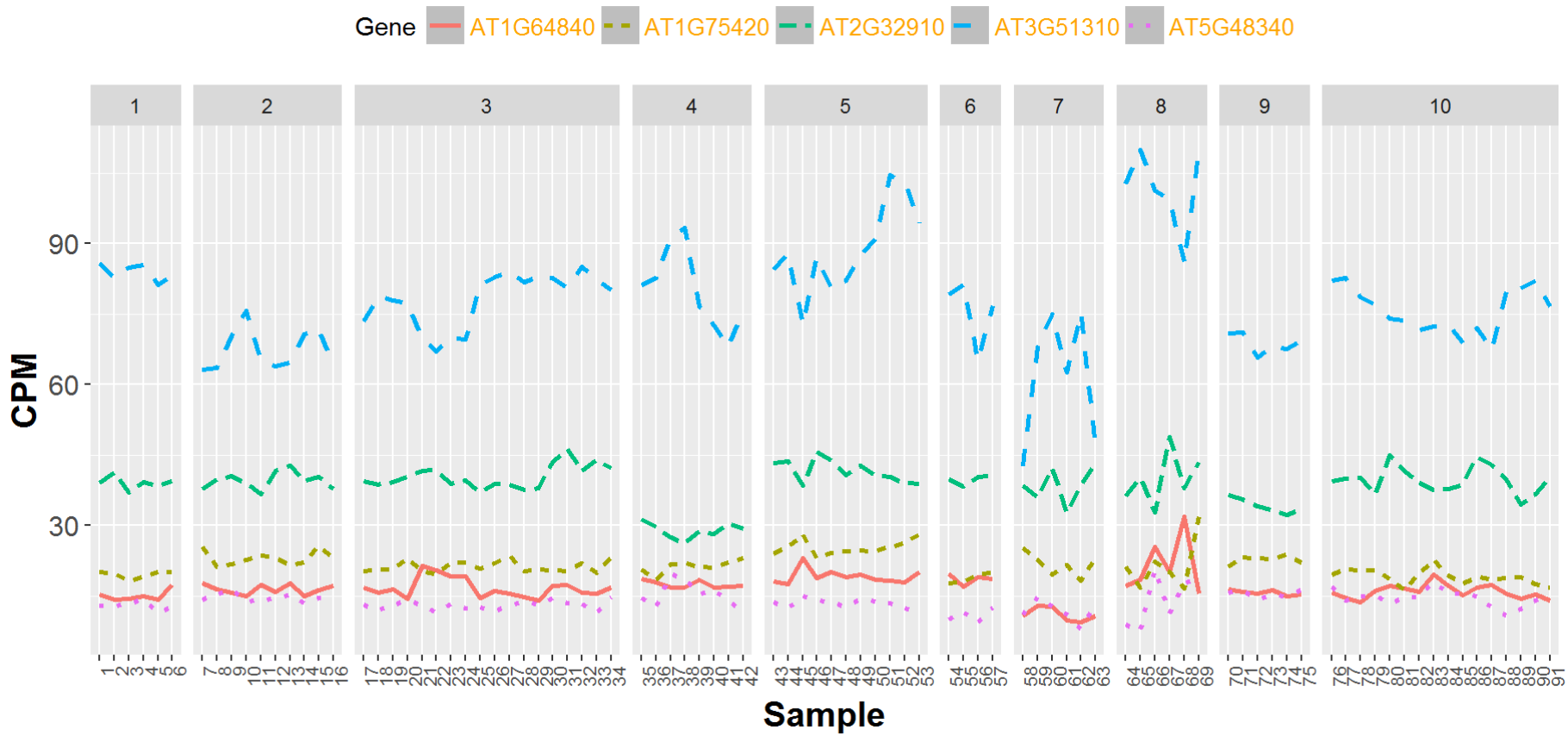
Facets display subsets of the dataset in different panels.

- `facet_grid` Lay out panels in a grid
- `facet_wrap` Wrap a 1-dimensional ribbon of panles into 2-dimensional
- ...

```
s2 + facet_wrap(~Gene, scales = "free")
```



```
s2 + facet_grid(~lab, scales = "free", space = "free")
```

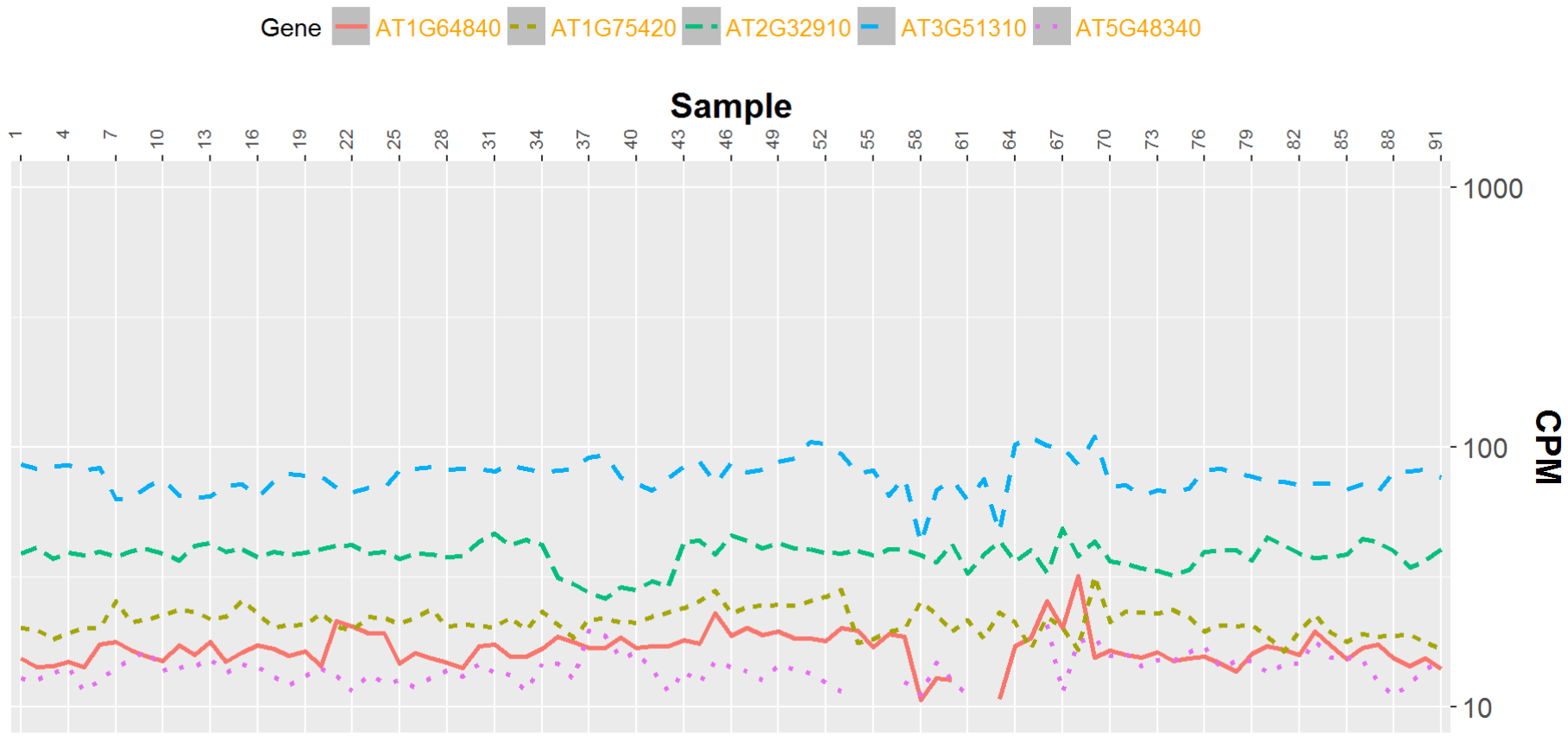


6. Scales

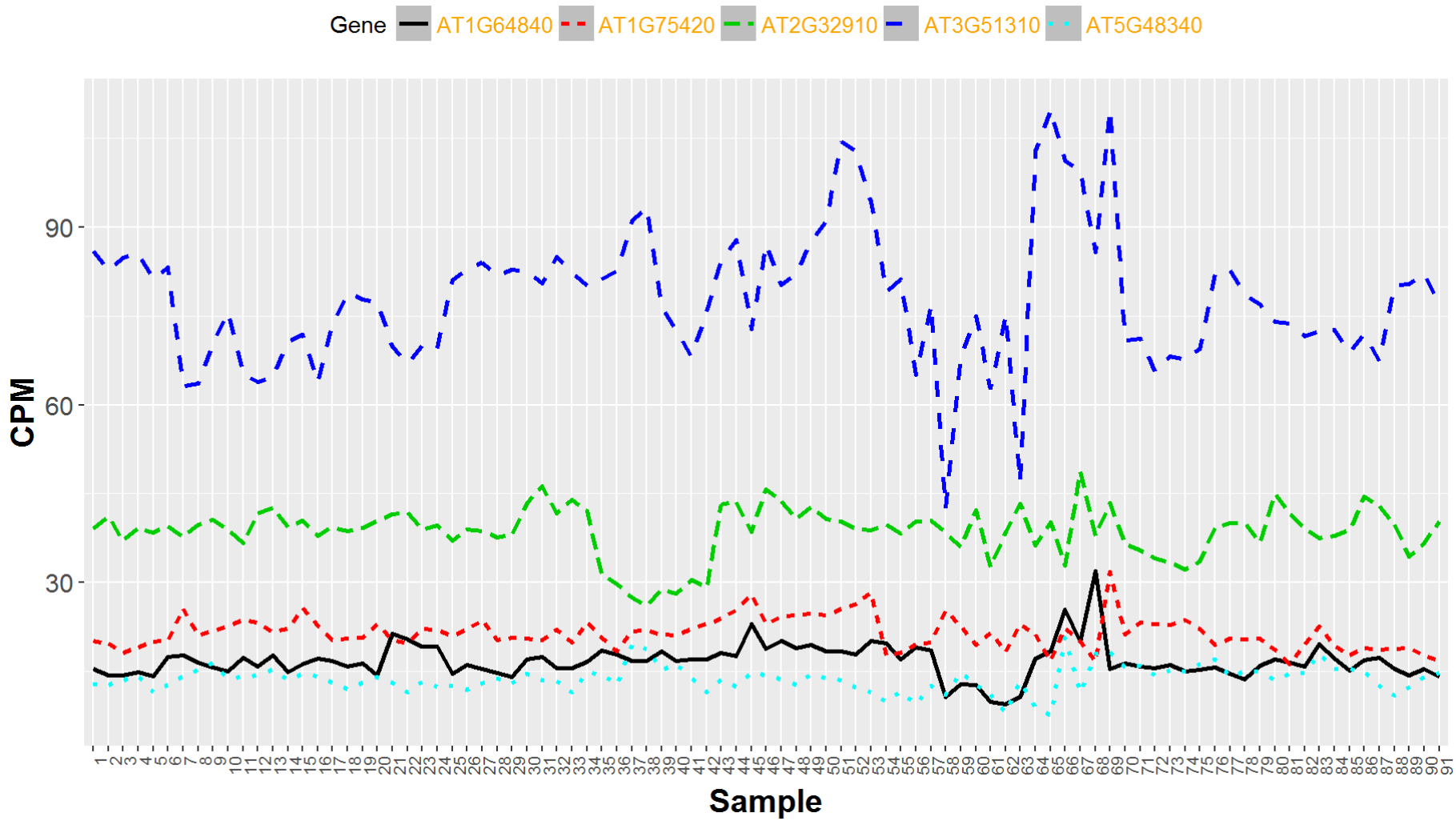
Scales control the mapping between data and aesthetics.

- `lims` set the axis limits.
- `scale_manual` Create your own discrete scale.
- `guide` set guides for each scale.
- ...

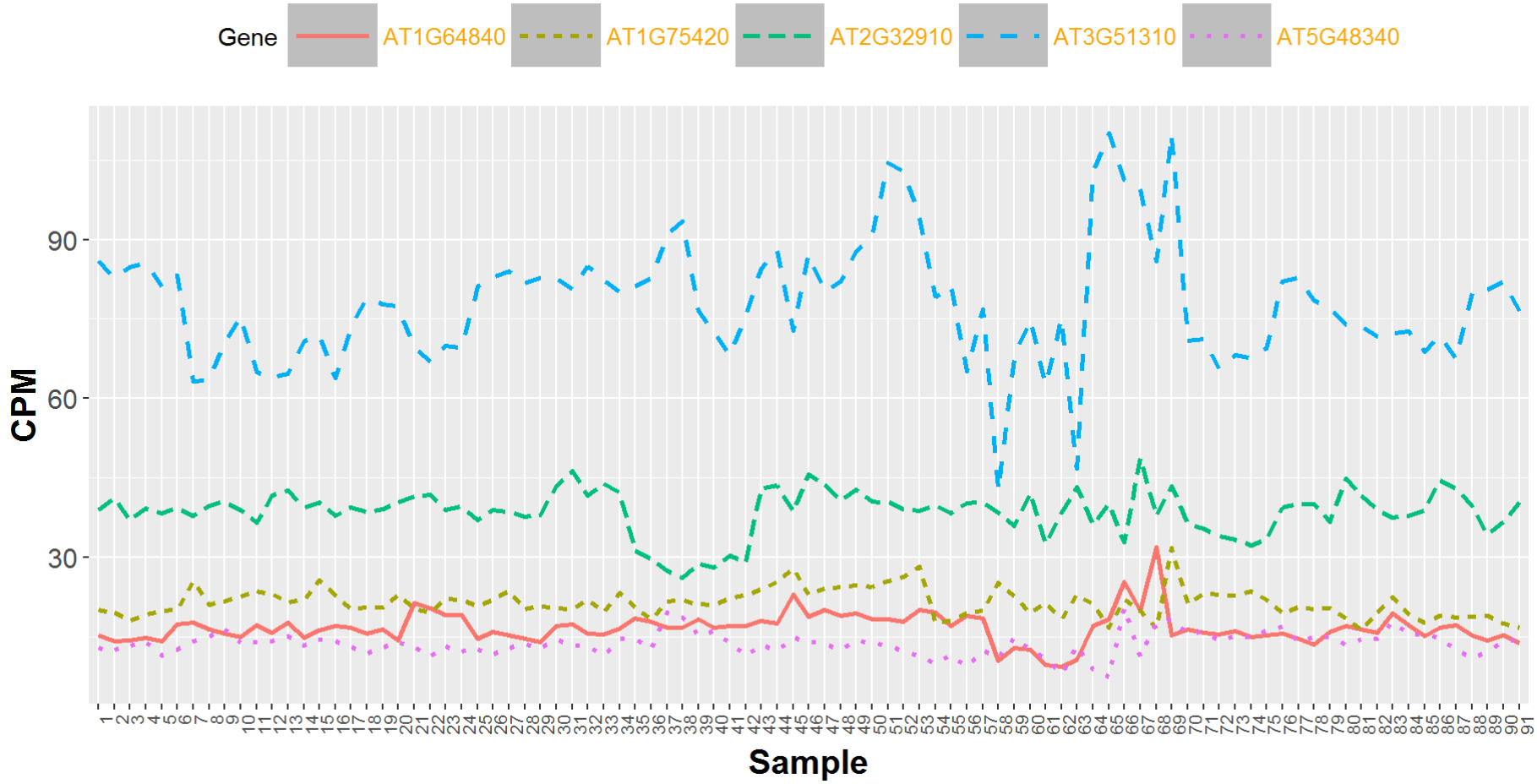
```
s2 + scale_x_discrete(breaks = seq(1,91, 3), position = "top") +
  scale_y_log10(limits = c(10, 1000), position ="right")
```



```
s2 + scale_color_manual(values = c(1:5) )
```

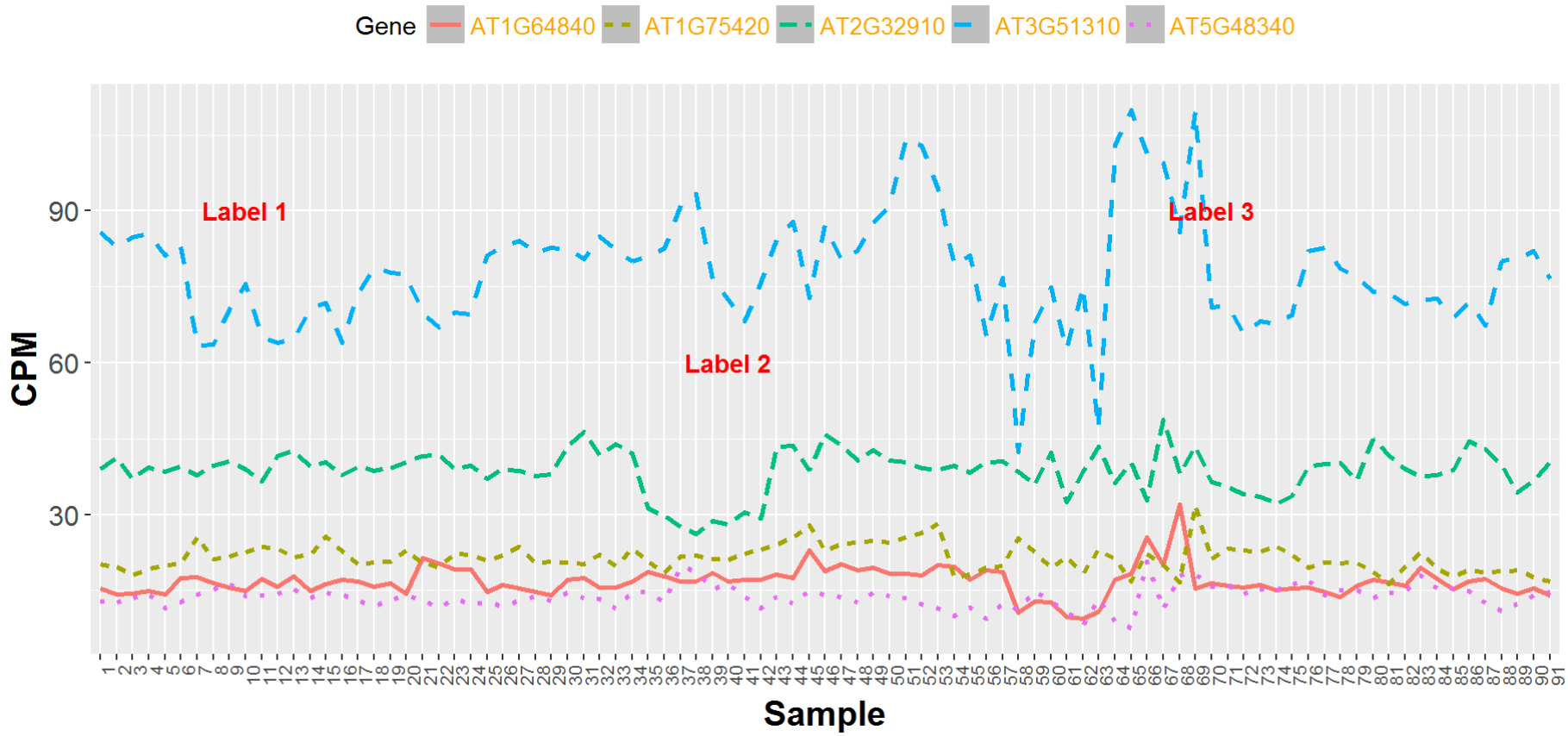


```
s2 + guides(linetype=guide_legend(keywidth = 2.8, keyheight = 2))
```

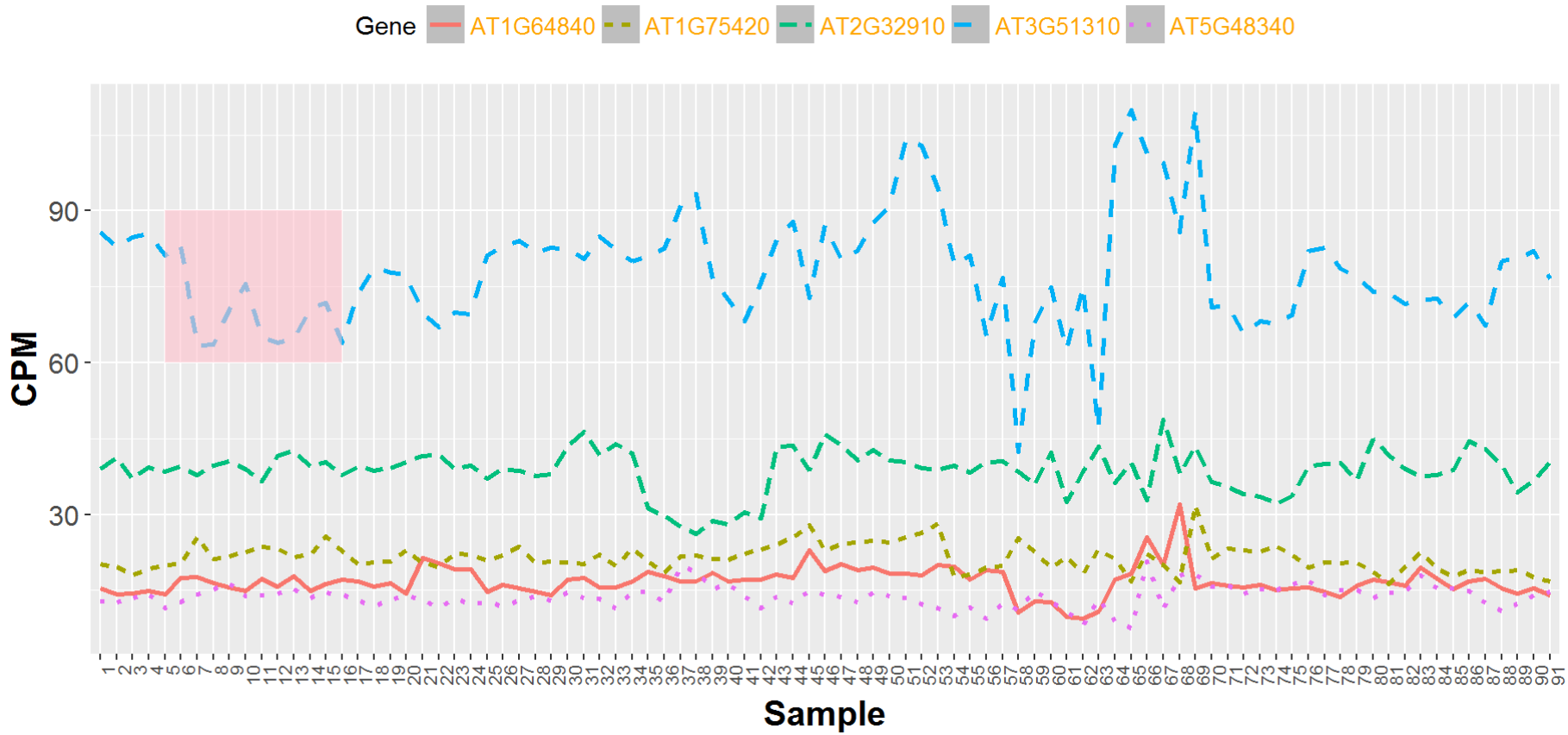


7. Annotation

```
s2 + annotate(geom = "text", x = c(10, 40, 70), y = c(90, 60, 90),
    label = c("Label 1", "Label 2", "Label 3"), color = "red", fontface= "bold")
```

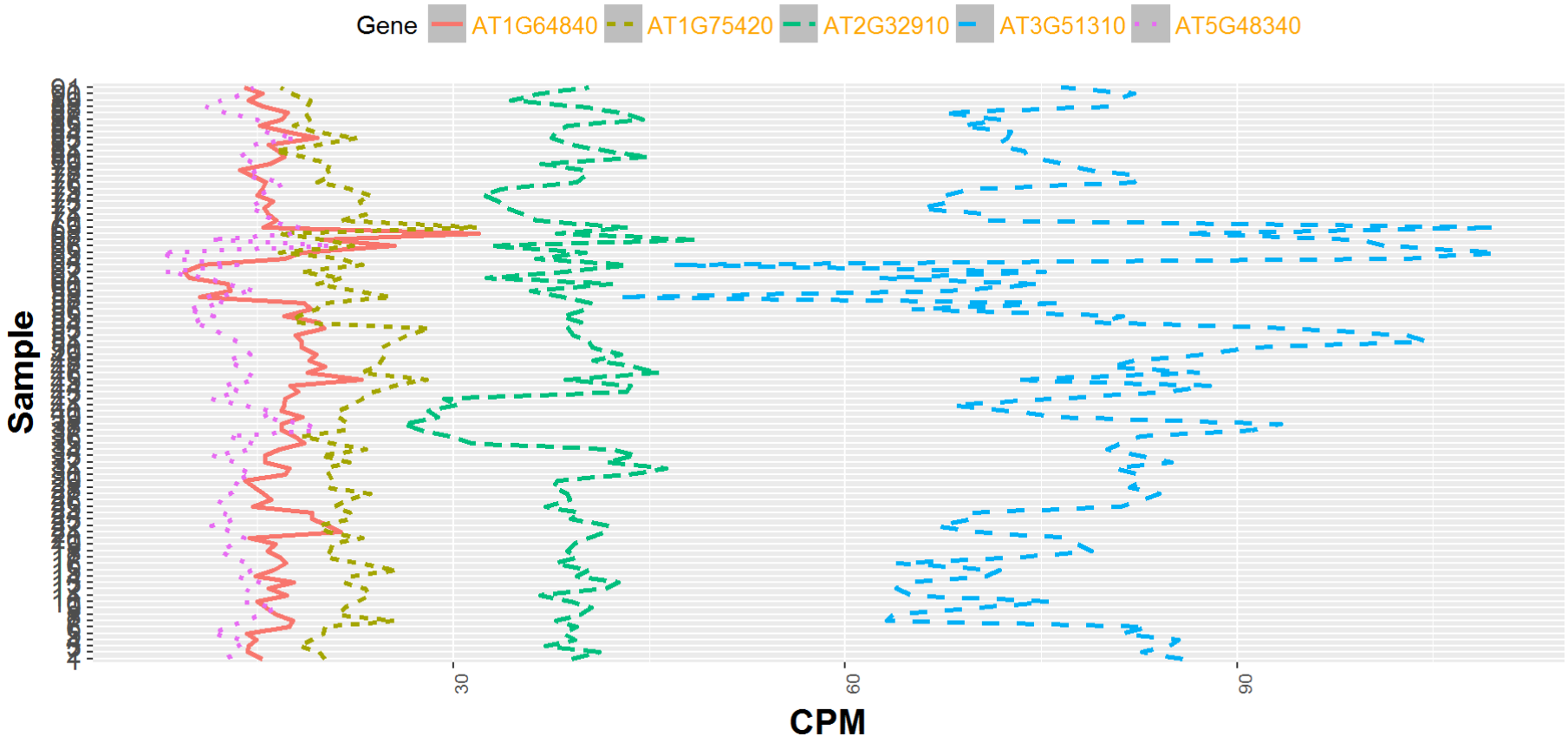


```
s2 + annotate(geom = "rect", xmin = 5, ymin = 60, xmax = 16, ymax = 90,
              fill = "pink", alpha = 0.6)
```



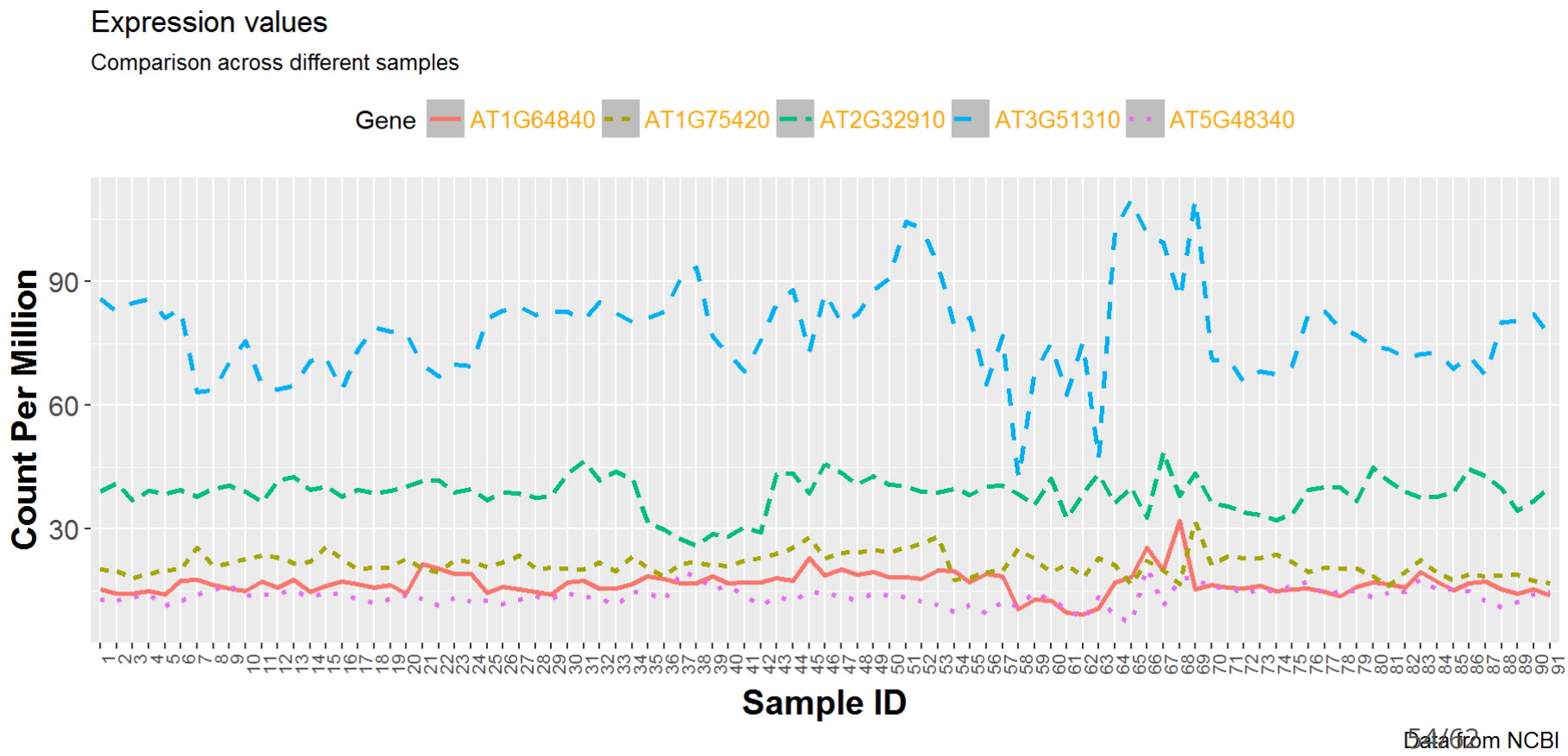
8. Coordinate system

```
s2 + coord_flip()
```

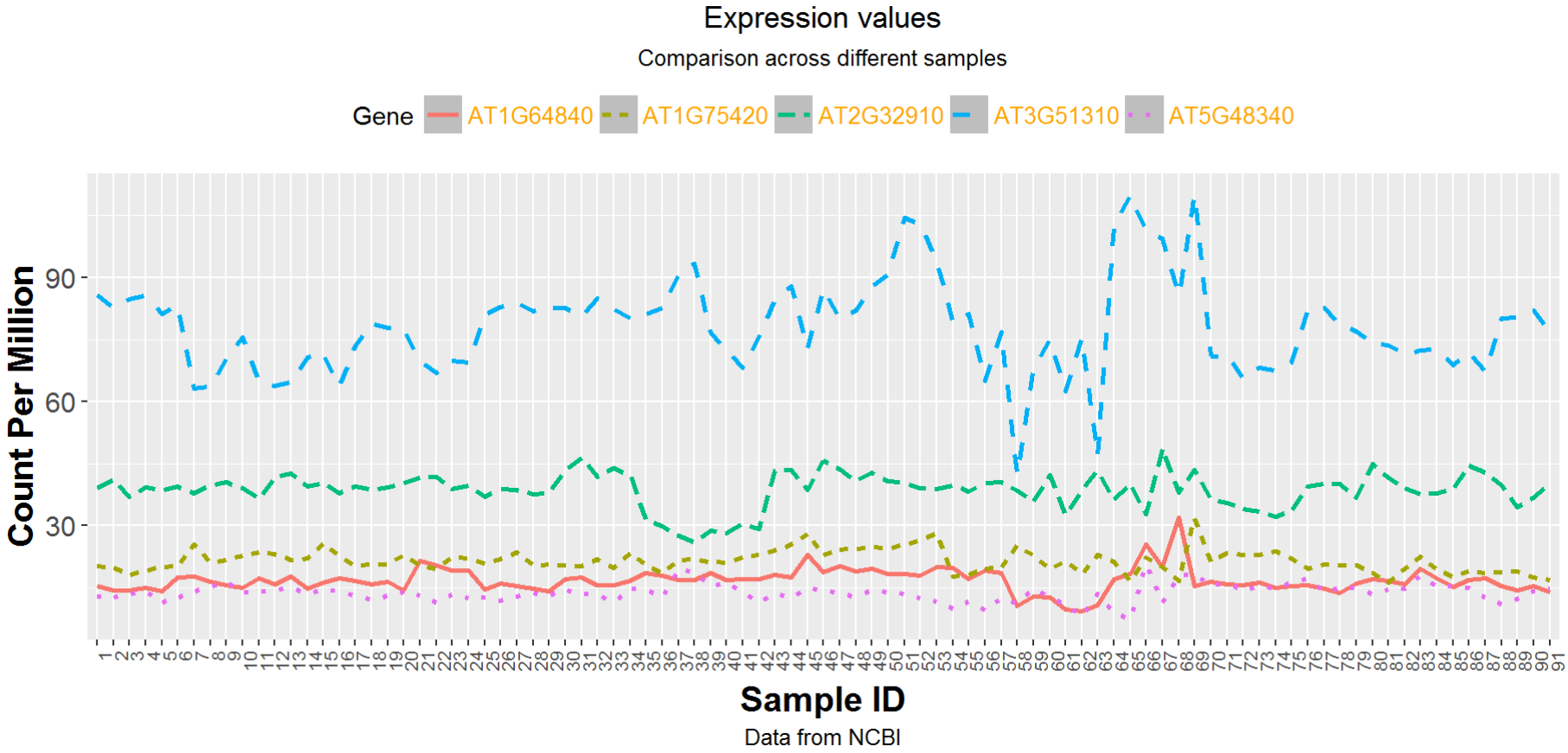


Title

```
s3 <- s2 + labs(x = "Sample ID", y = "Count Per Million", title = "Expression values",
        subtitle = "Comparison across different samples", caption = "Data from NCBI")
print(s3)
```



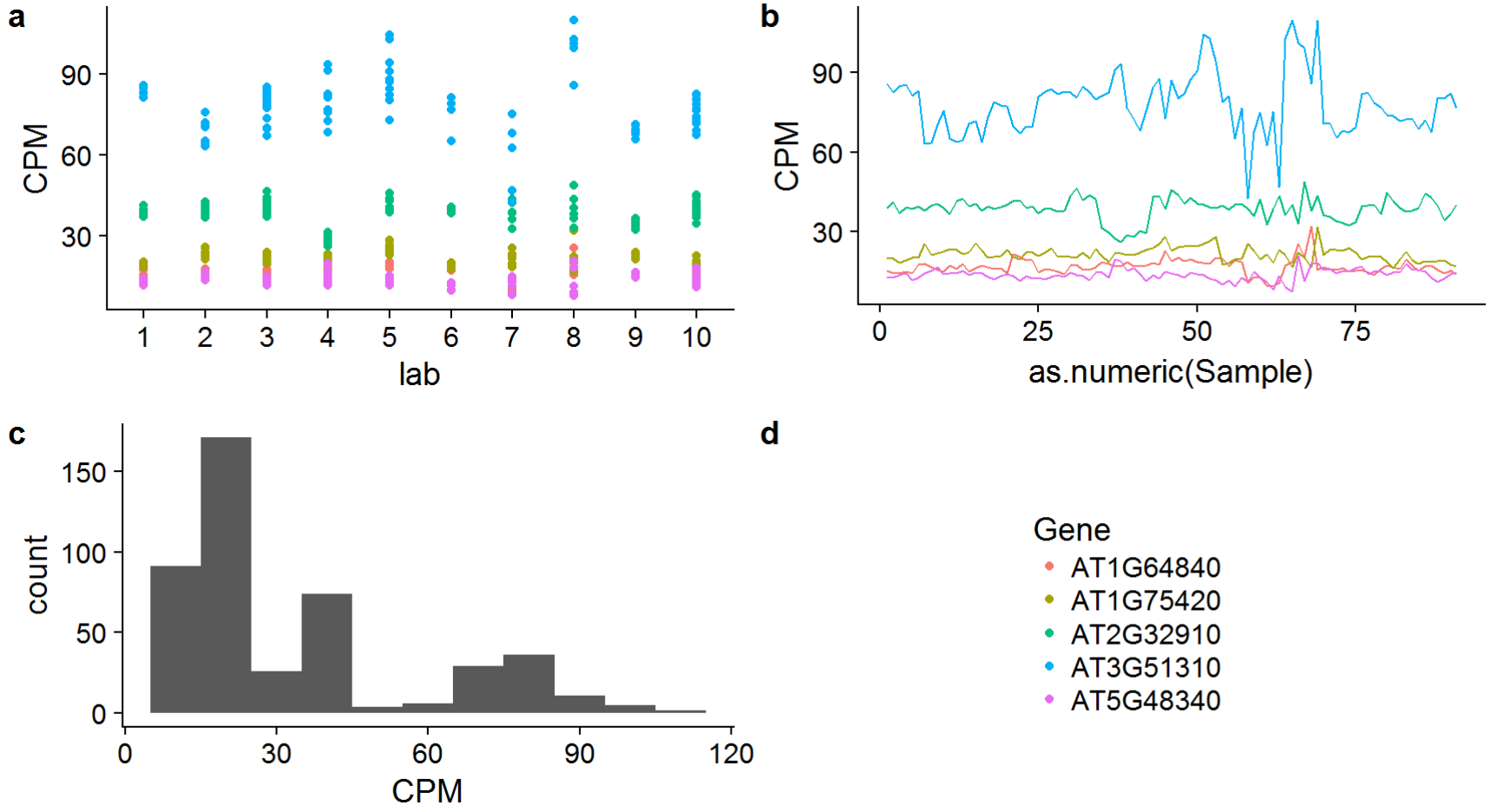
```
s3 + theme(plot.title = element_text(hjust = 0.5),
           plot.subtitle = element_text(hjust = 0.5),
           plot.caption = element_text(hjust = 0.5))
```



Arranging graphs into a grid

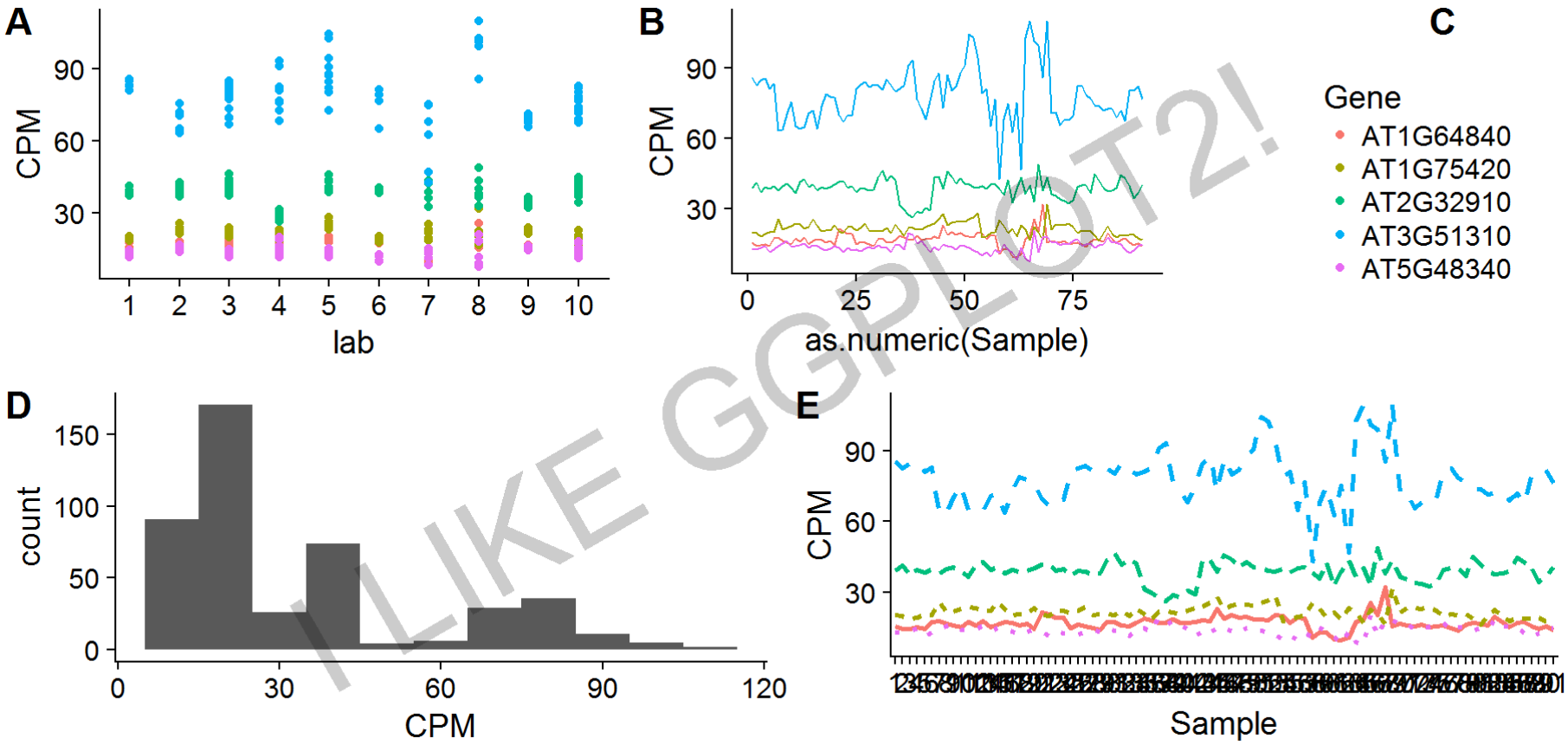
```
library(cowplot)

f0 <- ggplot(dat, aes(x = lab, y = CPM, color = Gene)) + geom_point()
f1 <- f0 + theme(legend.position = "none")
f2 <- ggplot(dat, aes(x = as.numeric(Sample), y = CPM, col = Gene)) +
  geom_line() + theme(legend.position = "none")

f3 <- ggplot(dat, aes(x= CPM)) + geom_histogram(binwidth = 10)

f4 <- get_legend(f0) # extract the legend
f5 <- s1 + theme(legend.position = "none")
```

```
plot_grid(f1, f2, f3, f4, labels = c("a", "b", "c", "d"),ncol =2, nrow = 2)
```



```
f6 <- ggdraw() + draw_plot(f1, x=0, y=0.5, width = 0.4, height = 0.5) +
  draw_plot(f2, x = 0.4, y = 0.5, width = .35, height = 0.5) +
  draw_plot(f4, 0.8, 0.5, 0.2, 0.5) + draw_plot(f3, 0, 0, 0.5, 0.5) +
  draw_plot(f5, 0.5, 0, 0.5, 0.5) +
  draw_plot_label(c("A", "B", "C", "D", "E"), c(0, 0.4, 0.9, 0, 0.5 ), c(1, 1, 1, 0.5, 0.5))
  draw_label("I LIKE GGPLOT2!", angle = 30, size = 60, alpha = 0.2)
```

```
print(f6)
```



Future: interactive graphics

1. `ggvis` http://ggvis.rstudio.com/
2. `shiny` http://shiny.rstudio.com/

Additional information

- Mailing list: http://groups.google.com/group/ggplot2
- Wiki: https://github.com/hadley/ggplot2/wiki
- Website: http://had.co.nz/ggplot2/
- StackOverflow: http://stackoverflow.com/questions/tagged/ggplot
- http://docs.ggplot2.org/current/.
- reach out to bin.zhuo@celerion.com if you have questions.

thank you!