Introduction to GGplot2

Anne Segonds-Pichon, Simon Andrews

v2021-09

Babraham Bioinformatics

Plotting figures and graphs with ggplot

- ggplot is the plotting library for tidyverse
  - Powerful
  - Flexible

- Follows the same conventions as the rest of tidyverse
  - Data stored in tibbles
  - Data is arranged in 'tidy' format
  - Tibble is the first argument to each function

Code structure of a ggplot graph

- Start with a call to `ggplot()`
  - Pass the tibble of data (normally via a pipe)
  - Say which columns you want to use via a call to `aes()`
- Say which graphical representation (geometry) you want to use
  - Points, lines, barplots etc
- Customise labels, colours annotations etc.

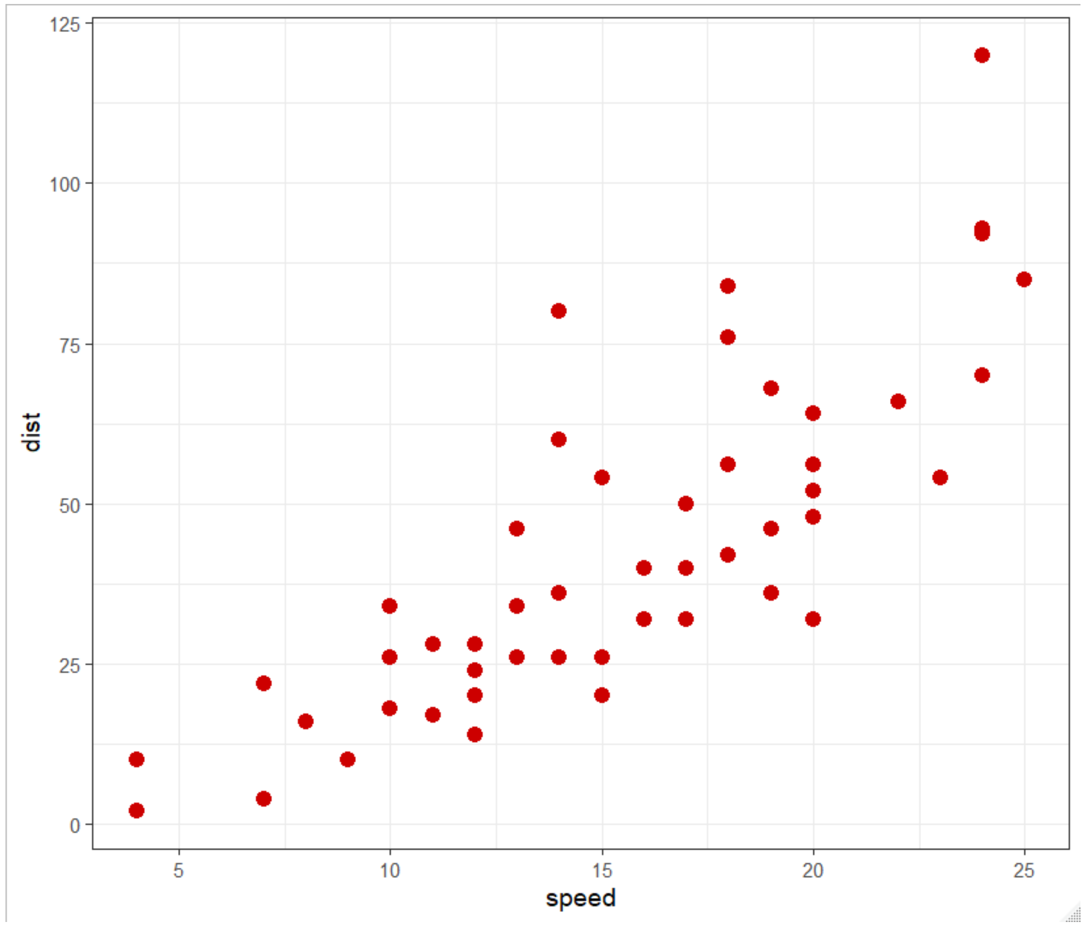
Geometries and Aesthetics

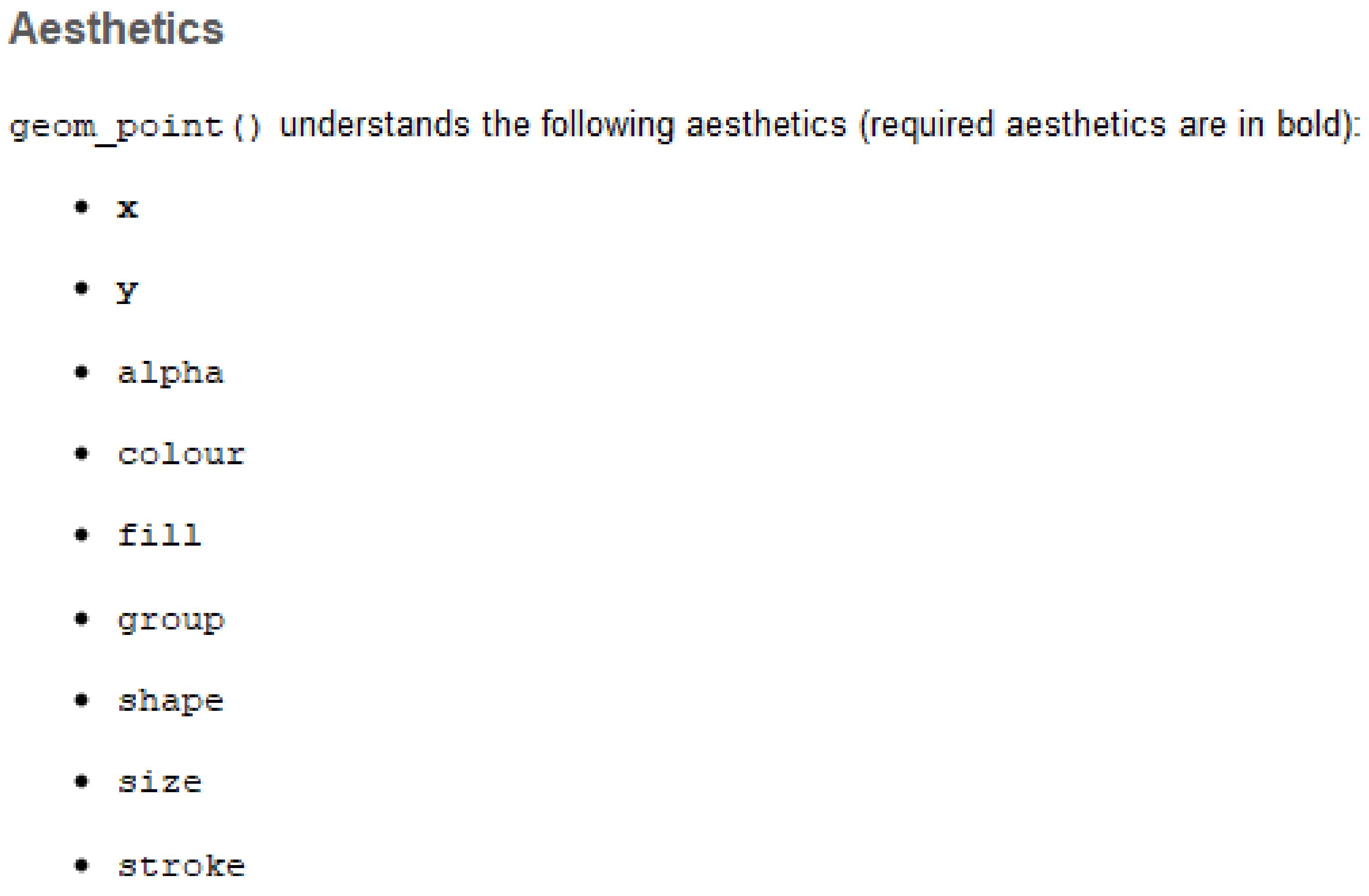
- Geometries are types of plot

`geom_point()`	Point geometry, (x/y plots, stripcharts etc)
`geom_line()`	Line graphs
`geom_boxplot()`	Box plots
`geom_col()`	Barplots
`geom_histogram()`	Histogram plots

- Aesthetics are graphical parameters which can be adjusted in a given geometry

Aesthetics for `geom_point()`





Aesthetics

`geom_point()` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `shape`
- `size`
- `stroke`

How do you define aesthetics

- Fixed values
  - Colour all points red
  - Make the points size 4

- Encoded from your data – called an *aesthetic mapping*
  - Colour according to genotype
  - Size based on the number of observations

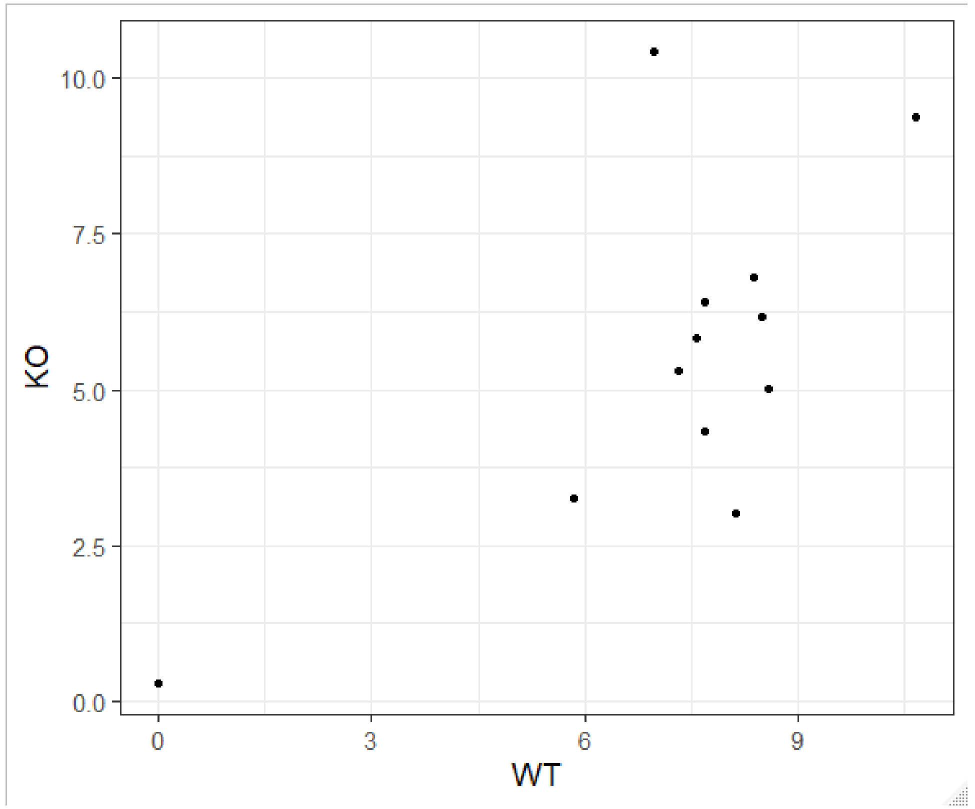
- Aesthetic mappings are set using the `aes()` function, normally as an argument to the `ggplot` function

```
data %>% ggplot(aes(x=weight, y=height, colour=genotype))
```

Our first plot…

```
ggplot(expression, aes(x=WT, y=KO)) + geom_point()
```

```
> expression
# A tibble: 12 x 4
   Gene        WT     KO pValue
   <chr>    <dbl>  <dbl>  <dbl>
 1 Mia1      5.83  3.24  0.1
 2 Snrpa     8.59  5.02  0.001
 3 Itpkc     8.49  6.16  0.04
 4 Adck4     7.69  6.41  0.2
 5 Numbl     8.37  6.81  0.1
 6 Ltbp4     6.96 10.4   0.001
 7 Shkbp1    7.57  5.83  0.1
 8 Spnb4    10.7   9.38  0.2
 9 Blvrb     7.32  5.29  0.05
10 Pgam1     0     0.285 0.5
11 Sertad3   8.13  3.02  0.0001
12 Sertad1   7.69  4.34  0.01
```



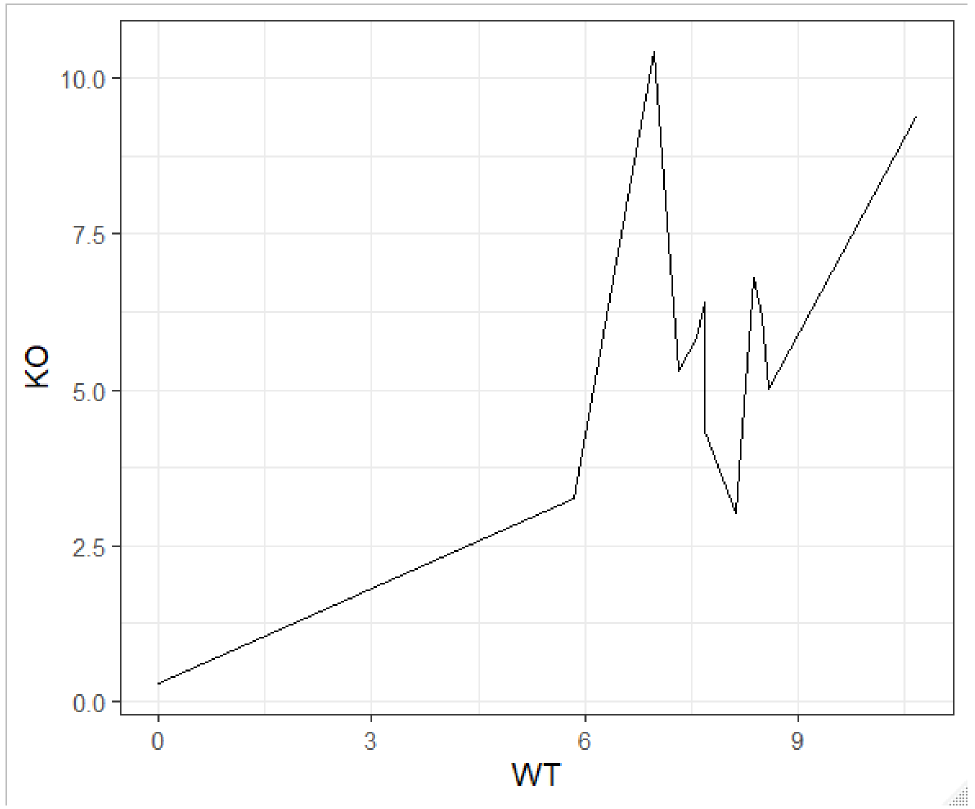
- Identify the tibble with the data you want to plot
- Decide on the geometry (plot type) you want to use
- Decide which columns will modify which aesthetic

- Call `ggplot(aes(…..))`
- Add a `geom_xxx` function call

Our second plot…

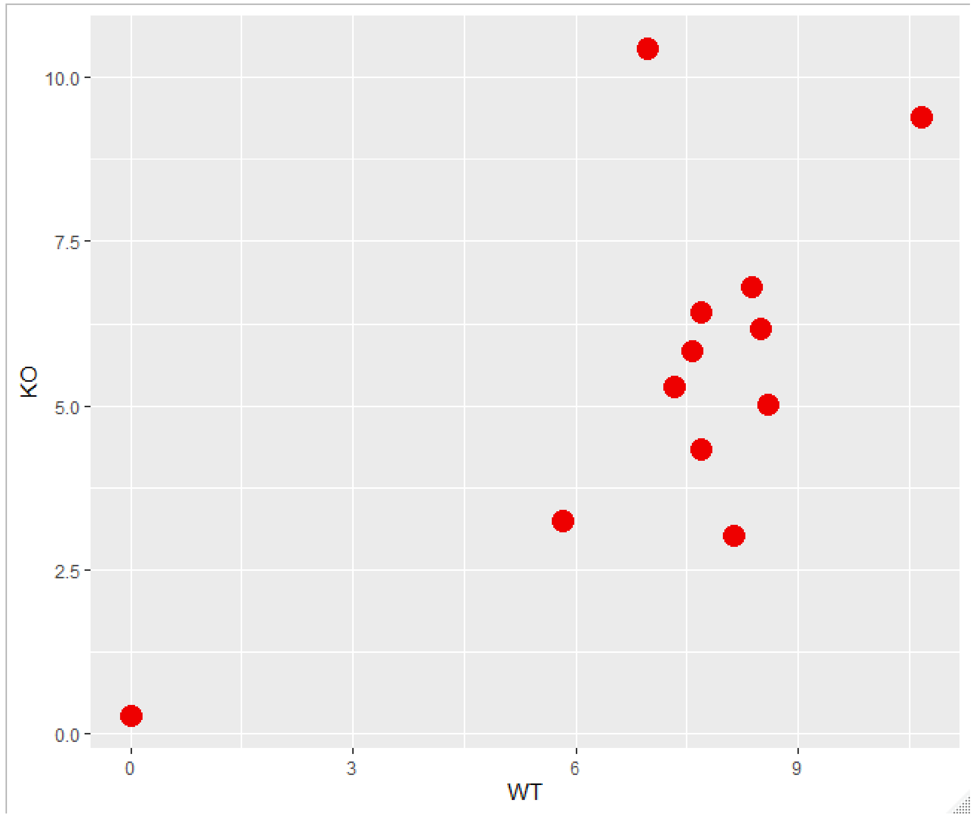
`ggplot(expression, aes(x=WT, y=KO)) + geom_line()`

```
> expression
# A tibble: 12 x 4
   Gene        WT     KO pValue
   <chr>    <dbl>  <dbl>  <dbl>
 1 Mia1      5.83  3.24  0.1
 2 Snrpa     8.59  5.02  0.001
 3 Itpkc     8.49  6.16  0.04
 4 Adck4     7.69  6.41  0.2
 5 Numbl     8.37  6.81  0.1
 6 Ltbp4     6.96 10.4   0.001
 7 Shkbp1    7.57  5.83  0.1
 8 Spnb4    10.7   9.38  0.2
 9 Blvrb     7.32  5.29  0.05
10 Pgam1     0     0.285 0.5
11 Sertad3   8.13  3.02  0.0001
12 Sertad1   7.69  4.34  0.01
```



Our third plot...

```
expression %>%
  ggplot (aes(x=WT, y=KO)) +
  geom_point(colour="red2", size=5)
```



Exercise 1

More Geometries

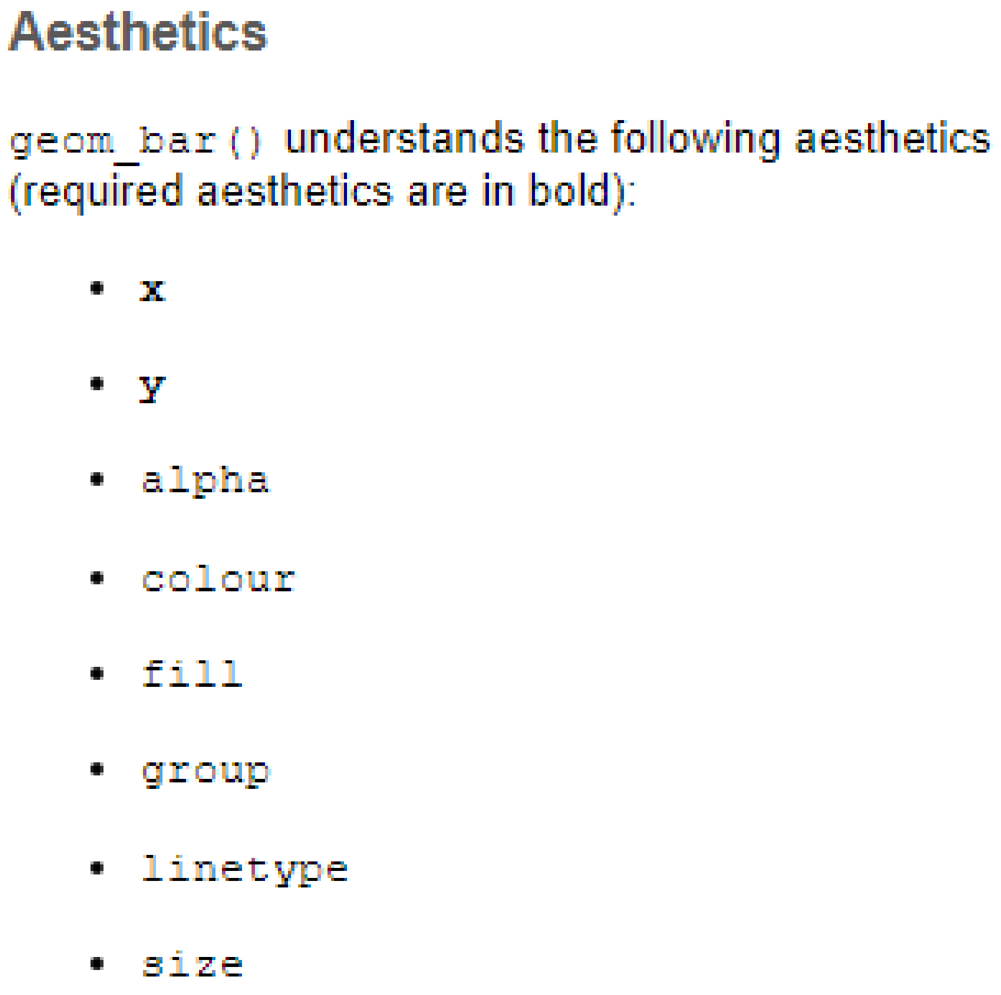
Other Geometries

- Barplots
  - geom_bar
  - geom_col
- Stripcharts
  - geom_jitter
- Distribution Summaries
  - geom_histogram
  - geom_density
  - geom_violin
  - geom_boxplot

Drawing a barplot (`geom_col()` or `geom_bar()`)

- Two different functions – depends on the nature of the data
- If your data has values which represents the height of the bars use `geom_col`
- If your data has individual values and you want the plot to either count them or calculate a quantitative summary (usually the mean) then use `geom_bar`
- Many geometries are "summarising geometries". They calculate one or more aesthetics for you.

Drawing a bar height barplot (`geom_col()`)



Aesthetics

`geom_bar()` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `linetype`
- `size`

- Plot the expression values for the WT samples for all genes

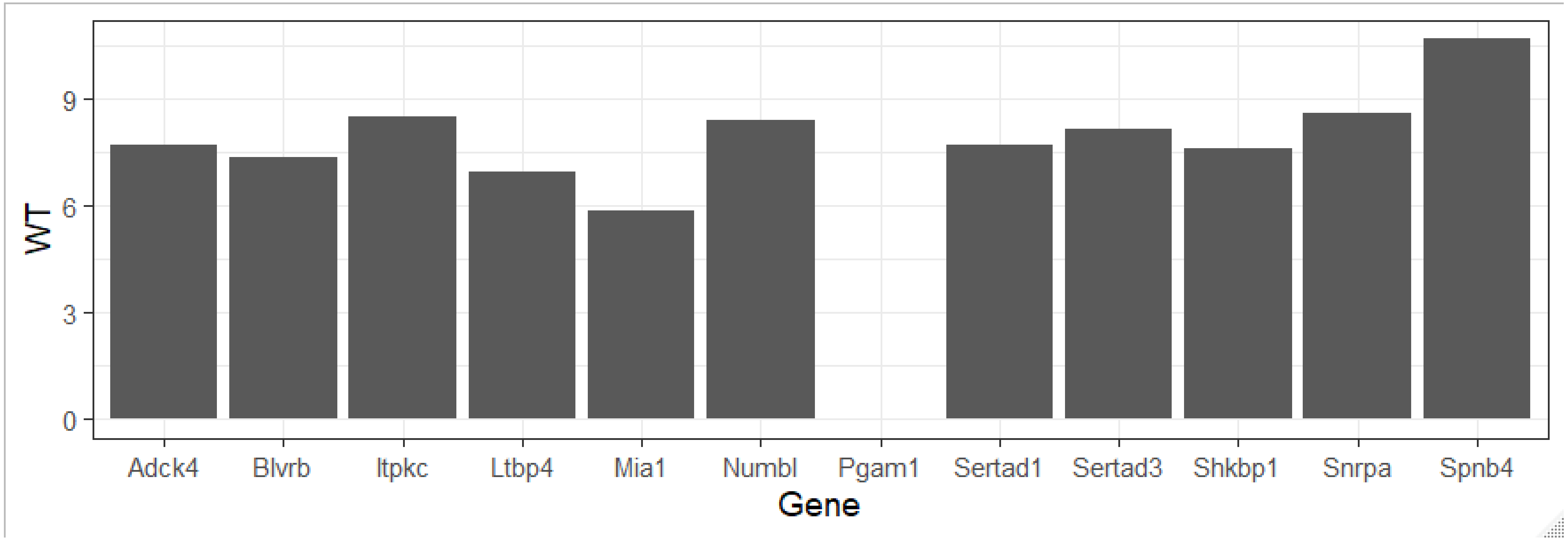
- What is your X?
- What is your Y?

```
> expression
# A tibble: 12 x 4
   Gene          WT      KO pValue
   <chr>      <dbl>   <dbl>  <dbl>
 1 Mia1        5.83   3.24   0.1
 2 Snrpa       8.59   5.02   0.001
```

A bar height barplot

```
expression %>%
  ggplot(aes(x=Gene, y=WT)) +
  geom_col()
```

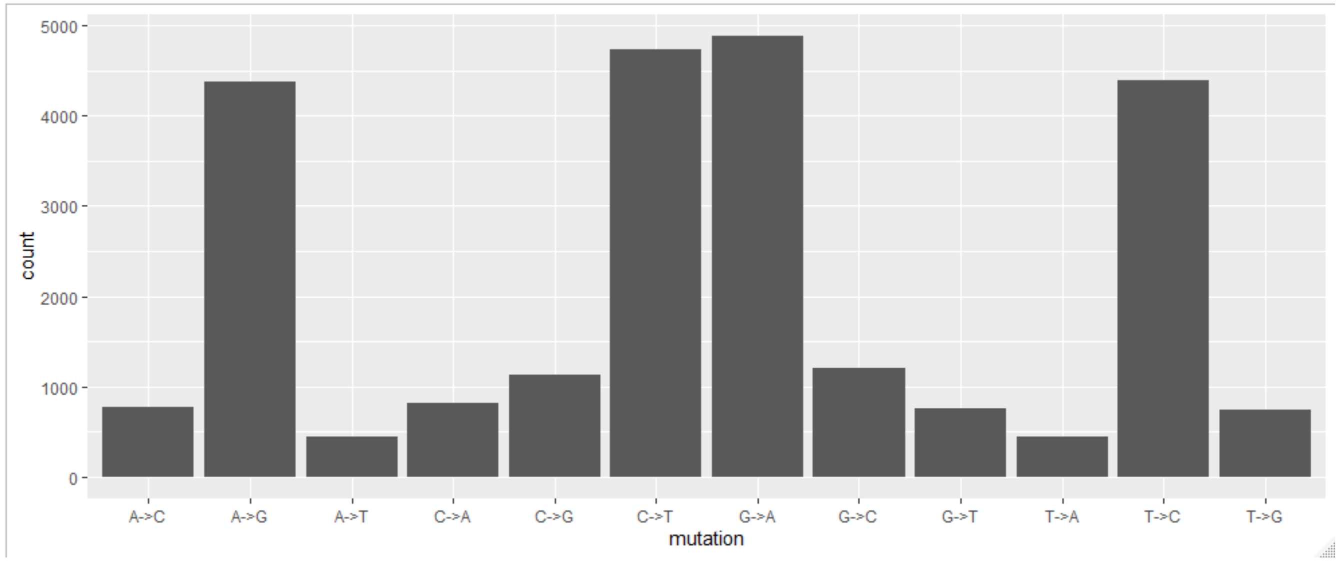
```
> expression
# A tibble: 12 x 4
   Gene        WT     KO pValue
   <chr>    <dbl>  <dbl>  <dbl>
 1 Mia1      5.83   3.24  0.1
 2 Snrpa     8.59   5.02  0.001
```



A count summary barplot (geom_bar)

```
mutation.plotting.data %>%
  ggplot(aes(x=mutation)) +
    geom_bar()
```

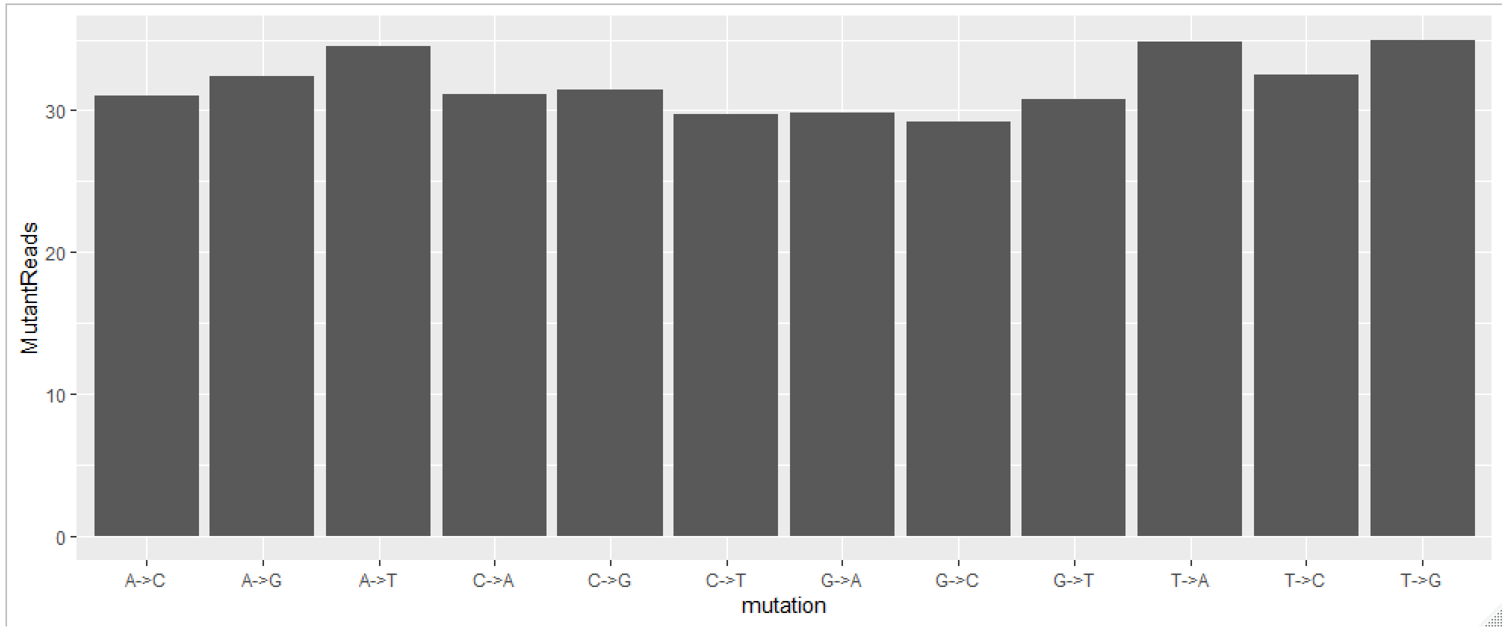
```
> mutation.plotting.data
# A tibble: 24,686 x 9
   CHR       POS dbSNP      mutation
   <chr>   <dbl> <chr>      <chr>
 1 1       69270 .          A->G
 2 1       69511 rs75062661 A->G
 3 1       69761 .          A->T
 4 1       69897 rs75758884 T->C
 5 1      877831 rs6672356  T->C
 6 1      881627 rs2272757  G->A
```



A mean summary barplot (geom_bar)

```
mutation.plotting.data %>%
  ggplot(aes(x=mutation, y=MutantReads)) +
  geom_bar(stat="summary", fun=mean)
```

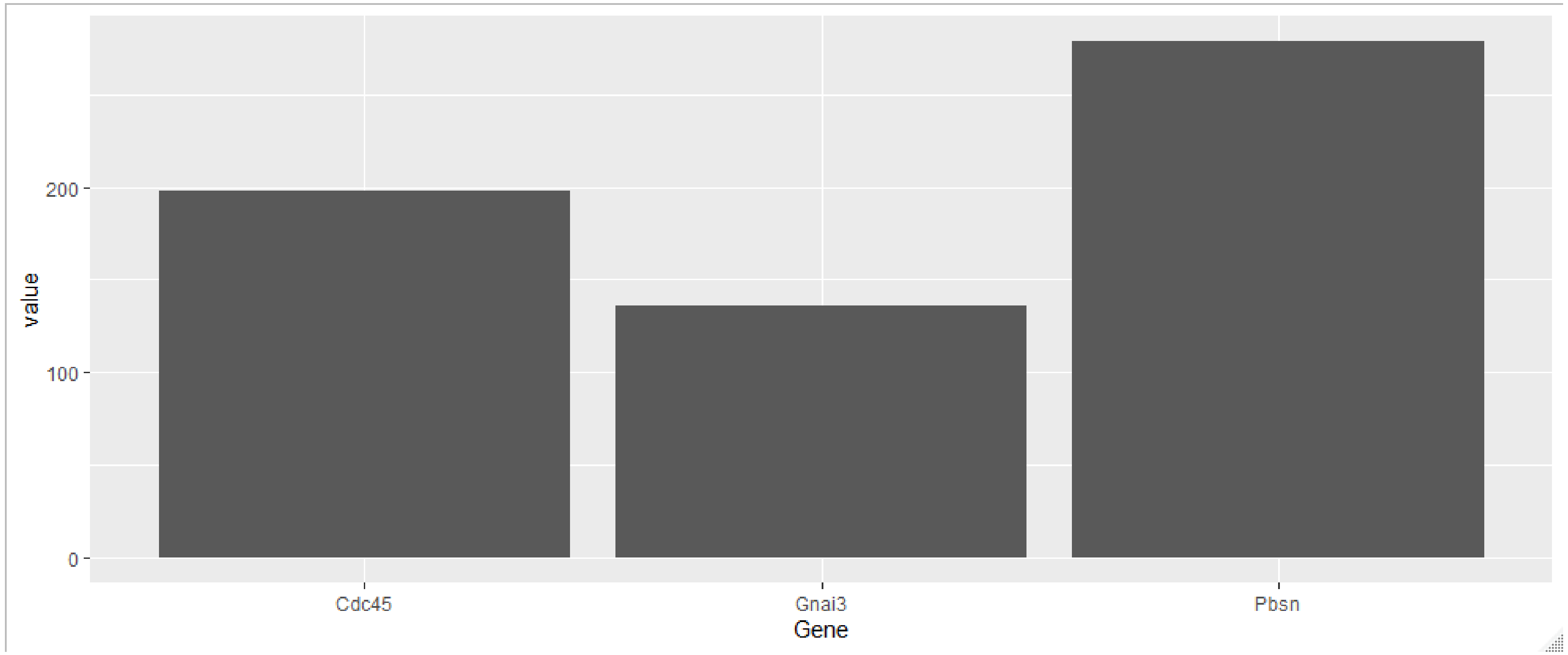
```
> mutation.plotting.data
# A tibble: 24,686 x 9
   CHR      POS mutation  MutantReads
   <chr>  <dbl> <chr>     <dbl>
 1 1      69270 A->G         3
 2 1      69511 A->G        24
 3 1      69761 A->T         8
 4 1      69897 T->C         3
 5 1     877831 T->C        10
 6 1     881627 G->A        52
 7 1     887801 A->G        47
 8 1     888639 T->C        23
 9 1     888659 T->C        17
10 1     889158 G->C        25
```



Stacked and Grouped Barplots

```
bar.group %>%
  ggplot(aes(x=Gene, y=value)) +
  geom_col()
```

```
> bar.group
# A tibble: 12 x 3
   Gene  genotype value
   <chr> <chr>    <dbl>
 1 Gnai3 WT        9.39
 2 Pbsn  WT       91.7
 3 Cdc45 WT       69.2
 4 Gnai3 WT       10.9
 5 Pbsn  WT       59.6
 6 Cdc45 WT       36.1
 7 Gnai3 KO       33.5
 8 Pbsn  KO       45.3
 9 Cdc45 KO       54.4
10 Gnai3 KO       81.9
11 Pbsn  KO       82.3
12 Cdc45 KO       38.1
```

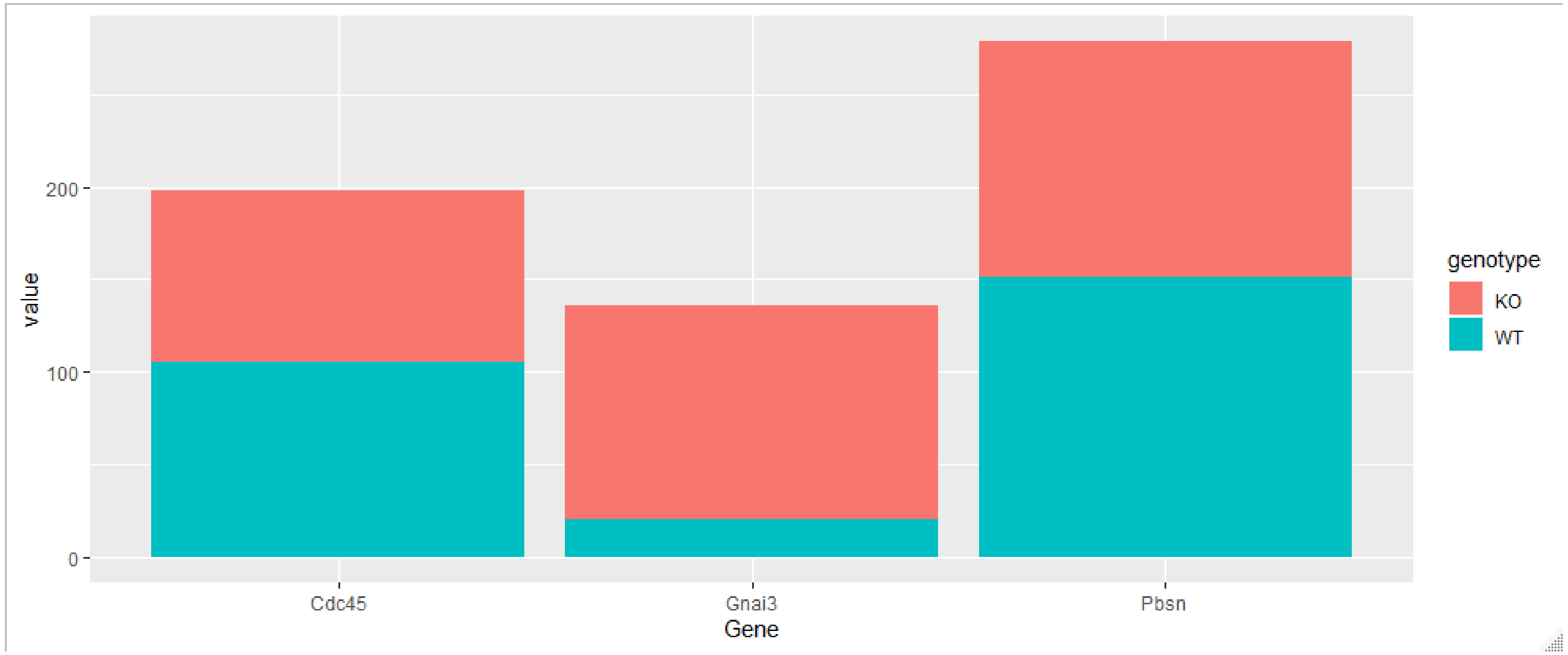


Sum of values

Stacked and Grouped Barplots

```
bar.group %>%
  ggplot(aes(x=Gene, y=value, fill=genotype)) +
  geom_col()
```

```
> bar.group
# A tibble: 12 x 3
   Gene  genotype value
   <chr> <chr>    <dbl>
 1 Gnai3 WT        9.39
 2 Pbsn  WT       91.7
 3 Cdc45 WT       69.2
 4 Gnai3 WT       10.9
 5 Pbsn  WT       59.6
 6 Cdc45 WT       36.1
 7 Gnai3 KO       33.5
 8 Pbsn  KO       45.3
 9 Cdc45 KO       54.4
10 Gnai3 KO       81.9
11 Pbsn  KO       82.3
12 Cdc45 KO       38.1
```

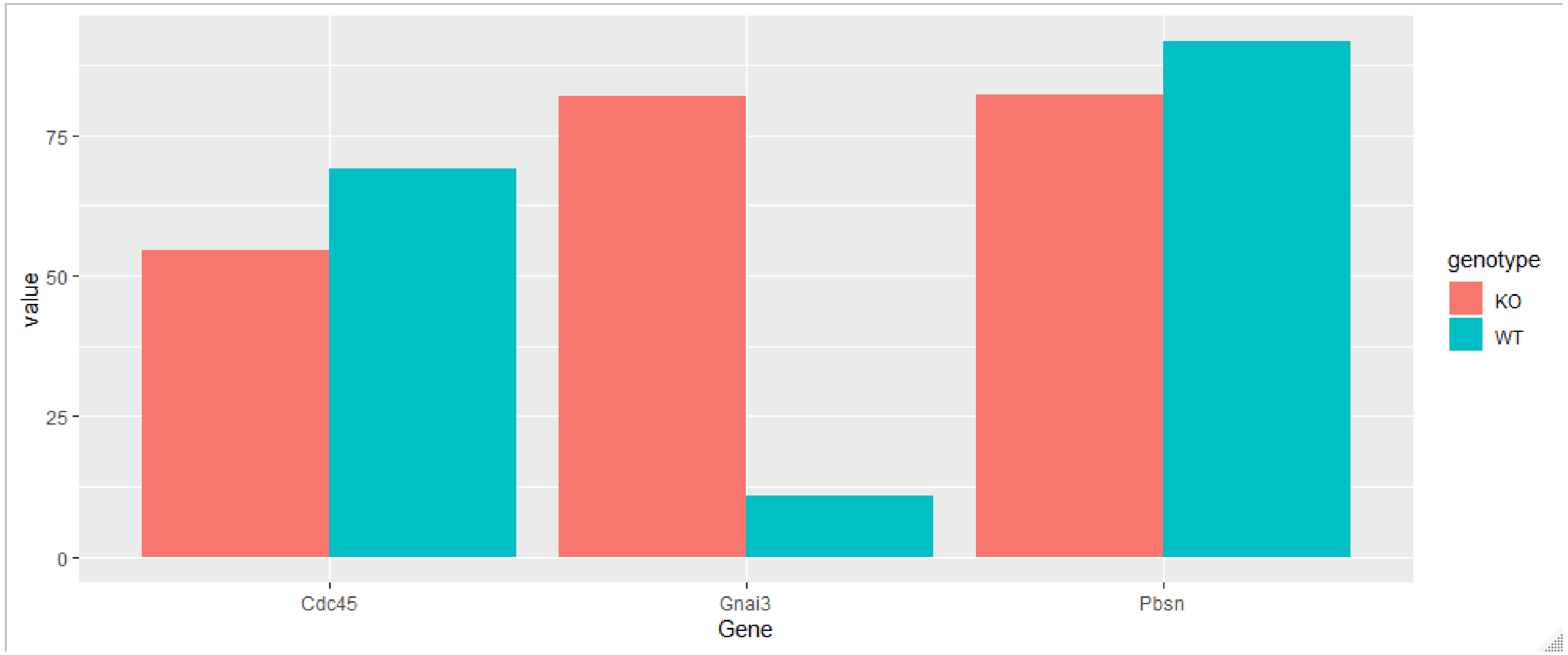


Stacked Sums

Stacked and Grouped Barplots

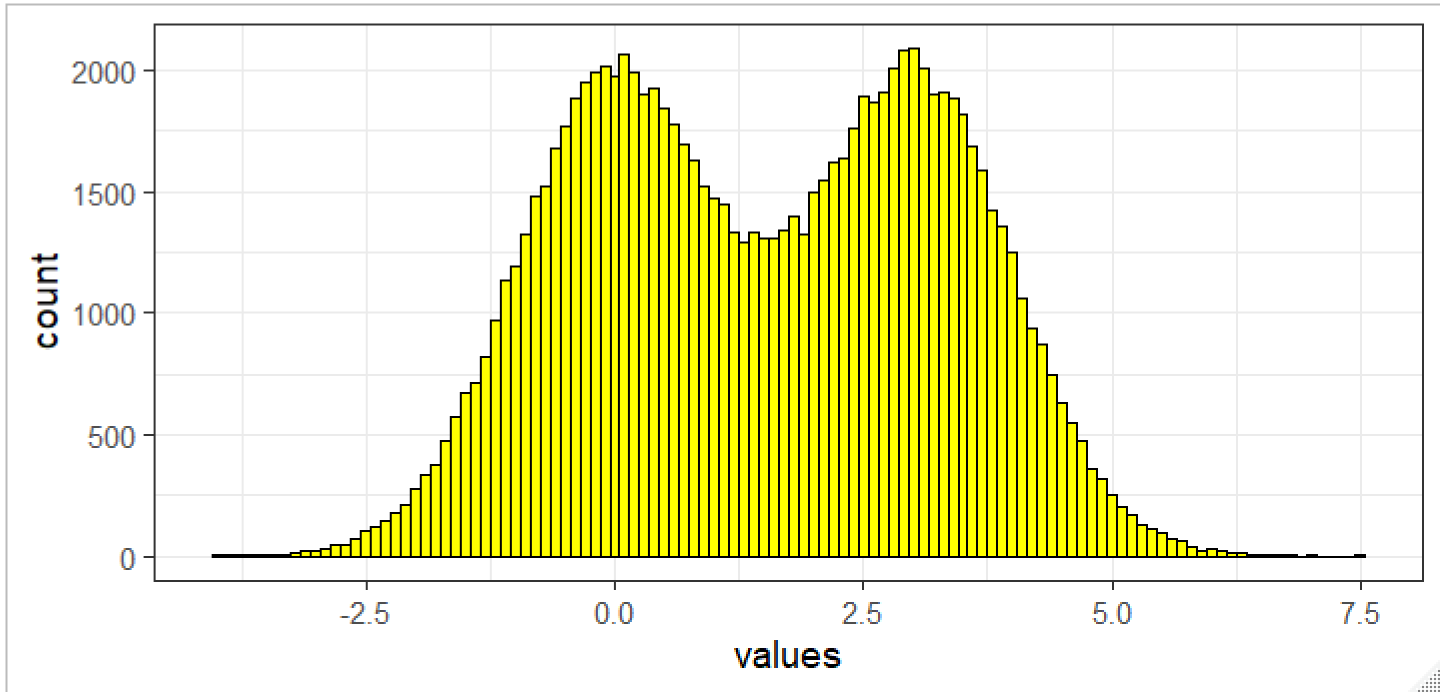
```
bar.group %>%
  ggplot(aes(x=Gene, y=value, fill=genotype)) +
  geom_col(position="dodge")
```

```
> bar.group
# A tibble: 12 x 3
   Gene  genotype value
   <chr> <chr>    <dbl>
 1 Gnai3 WT        9.39
 2 Pbsn  WT       91.7
 3 Cdc45 WT       69.2
 4 Gnai3 WT       10.9
 5 Pbsn  WT       59.6
 6 Cdc45 WT       36.1
 7 Gnai3 KO       33.5
 8 Pbsn  KO       45.3
 9 Cdc45 KO       54.4
10 Gnai3 KO       81.9
11 Pbsn  KO       82.3
12 Cdc45 KO       38.1
```



Individual values

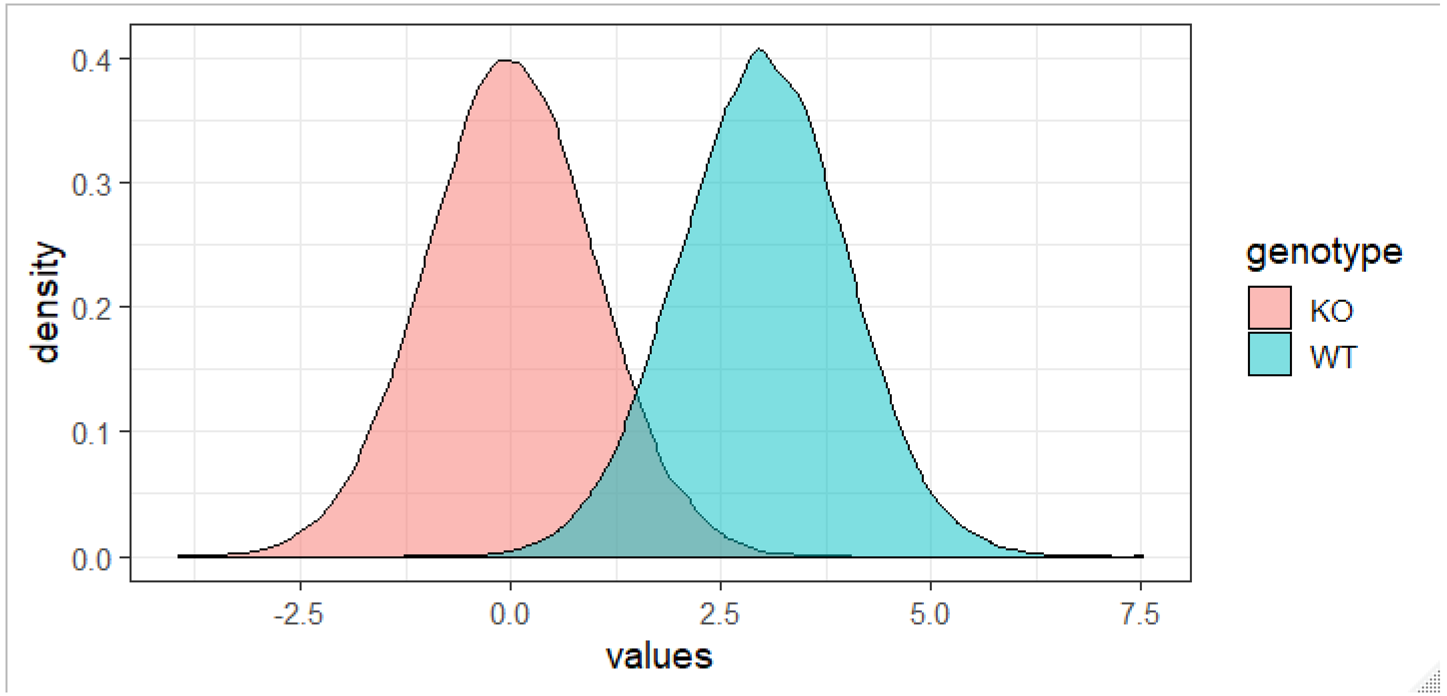
Plotting distributions - histograms



```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90  KO
 2  2.39  WT
 3  4.32  KO
 4  2.94  KO
 5  0.728 WT
 6 -0.280 WT
 7  0.337 WT
 8 -1.31  WT
 9  1.55  WT
10  1.86  KO
```

```
many.values %>%
  ggplot(aes(x=values)) +
  geom_histogram(binwidth = 0.1, fill="yellow", colour="black")
```

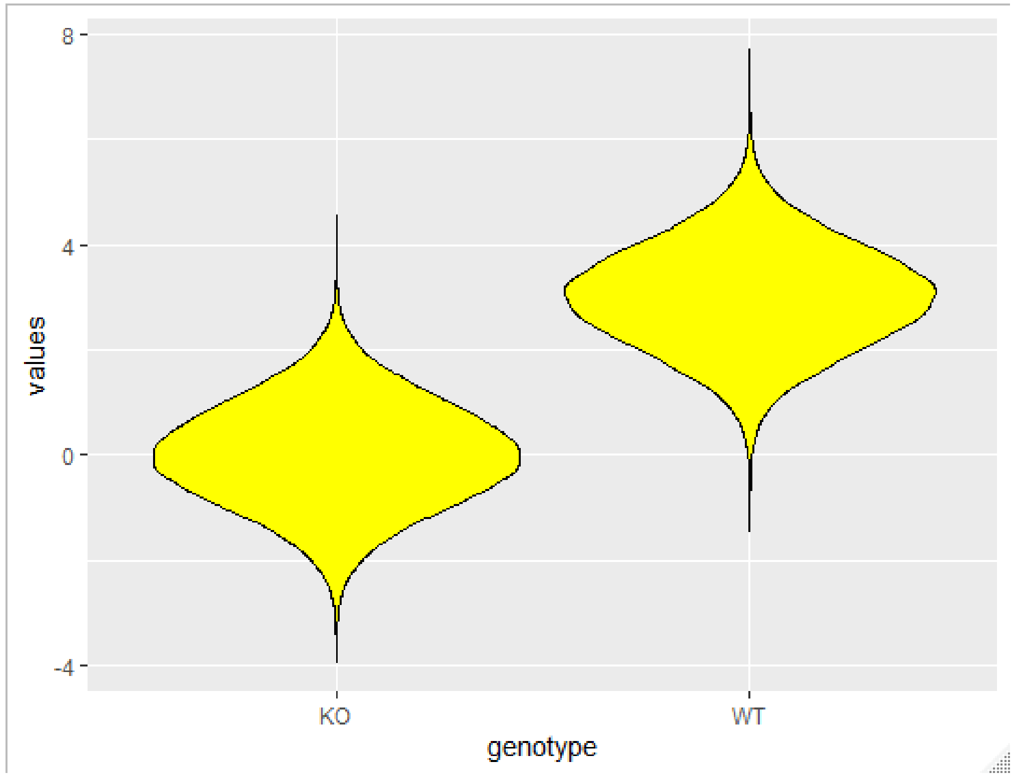
Plotting distributions - density



```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90  KO
 2  2.39  WT
 3  4.32  KO
 4  2.94  KO
 5  0.728 WT
 6 -0.280 WT
 7  0.337 WT
 8 -1.31  WT
 9  1.55  WT
10  1.86  KO
```

```
many.values %>%
  ggplot(aes(x=values, fill=genotype)) +
  geom_density(colour="black", alpha=0.5)
```

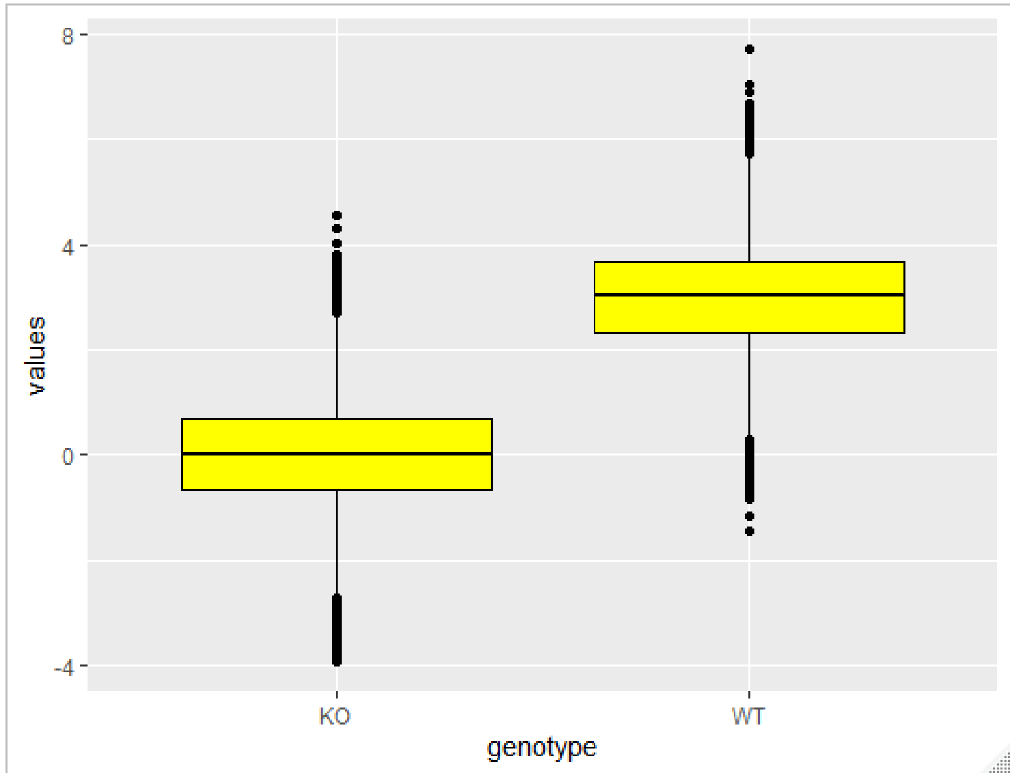
Plotting distributions – violin plots



```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90  KO
 2  2.39  WT
 3  4.32  KO
 4  2.94  KO
 5  0.728 WT
 6 -0.280 WT
 7  0.337 WT
 8 -1.31  WT
 9  1.55  WT
10  1.86  KO
```

```
many.values %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_violin(colour="black", fill="yellow")
```

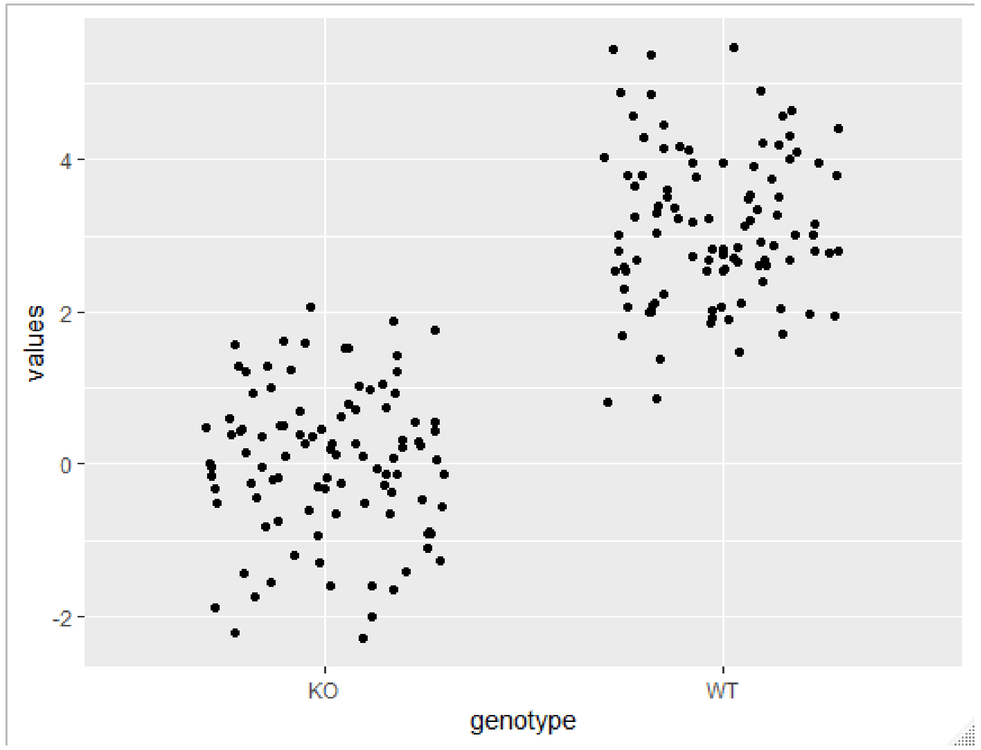
Plotting distributions – boxplots



```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90  KO
 2  2.39  WT
 3  4.32  KO
 4  2.94  KO
 5  0.728 WT
 6 -0.280 WT
 7  0.337 WT
 8 -1.31  WT
 9  1.55  WT
10  1.86  KO
```

```
many.values %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_boxplot(colour="black", fill="yellow")
```

Plotting distributions – stripcharts



```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90   KO
 2  2.39   WT
 3  4.32   KO
 4  2.94   KO
 5  0.728  WT
 6 -0.280  WT
 7  0.337  WT
 8 -1.31   WT
 9  1.55   WT
10  1.86   KO
```

```
many.values %>%
  group_by(genotype) %>%
  sample_n(100) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_jitter(height=0, width = 0.3)
```

Exercise 2

Annotation, Scaling and Colours

Babraham Bioinformatics

Titles and axis labels

- Can add calls to functions to set them individually
  - `ggtitle("Main title")`
  - `xlab("X axis")`
  - `ylab("Y axis")`

- Can set them all together with `labs()`
  - `title="Main title"`
  - `x="X axis"`
  - `y="Y axis"`

Changing scaling

- Alter the data before plotting
  - `mutate(value=log(value))`

- Alter the data whilst plotting
  - `ggplot(aes(x=log(value)))`

- Alter the scale of the plot
  - Add an option to adjust the scaling of the axis

Axis scaling options

- Transforming scales
  - `scale_x_log10()`
  - `scale_x_sqrt()`
  - `scale_x_reverse()`

  Equivalent `_y_` versions also exist

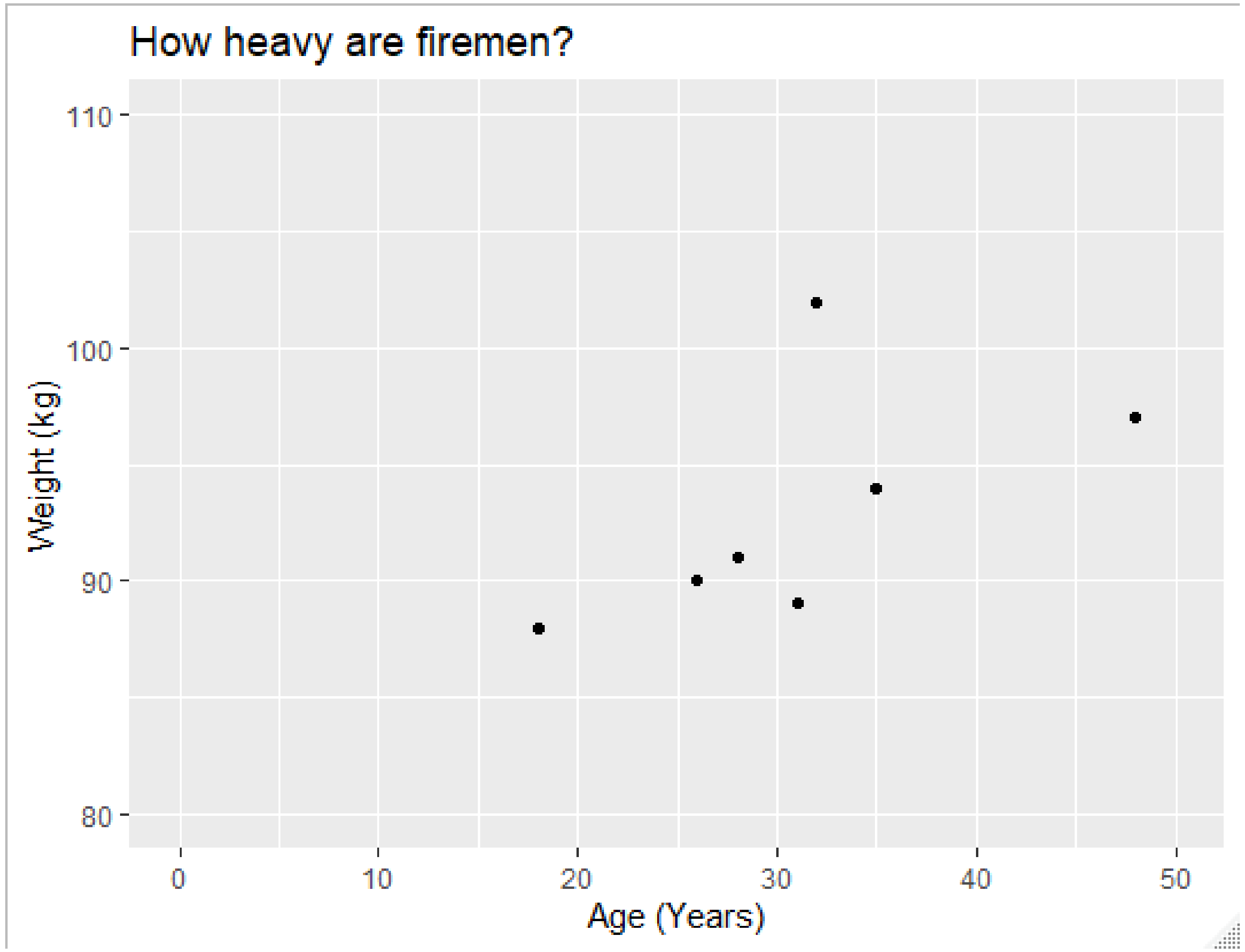
- Switching axes
  - `coord_flip()`

- Adjusting ranges
  - `scale_x_continuous()`
    - `limits=c(-5,5)`
    - `breaks=seq(from=-5,by=2,to=5)`
    - `minor_breaks`
    - `labels`

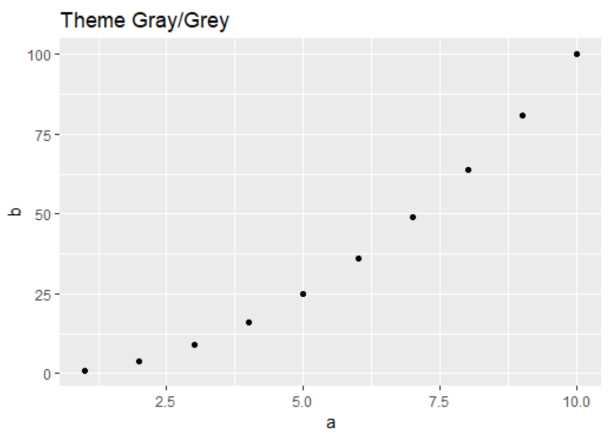
  - `coord_cartesian()`
    - `xlim=c(-5,5)`
    - `ylim=c(10,20)`

Annotation and scaling example

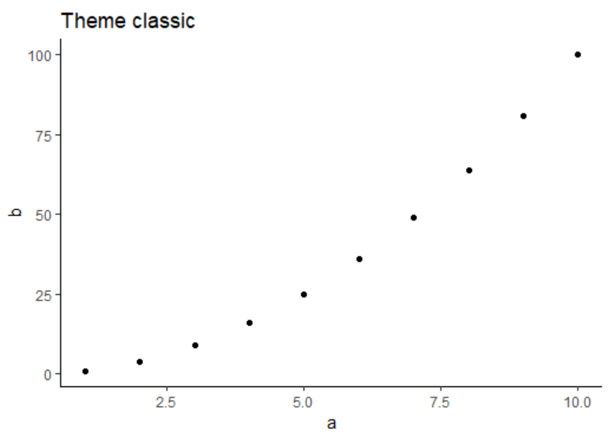
```
trumpton %>%
  ggplot(aes(x=Age, y=Weight))+
  geom_point() +

  xlab("Age (Years)")+
  ylab("Weight (kg)")+
  ggtitle("How heavy are firemen?")+

  coord_cartesian(
   xlim=c(0,50),
   ylim=c(80,110)
  )
```

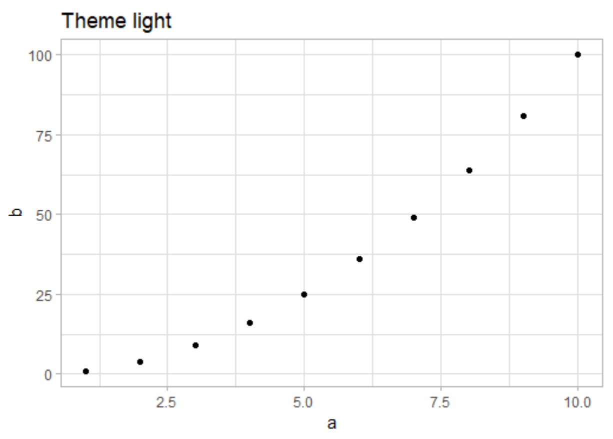


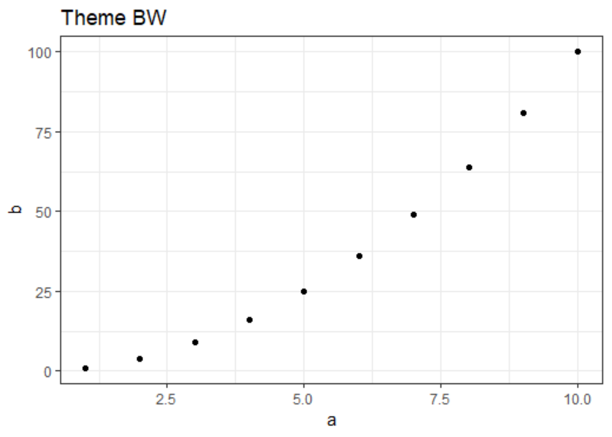
ggPlot Themes

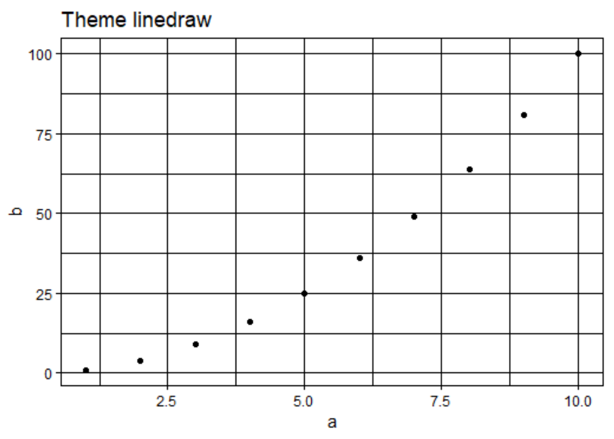


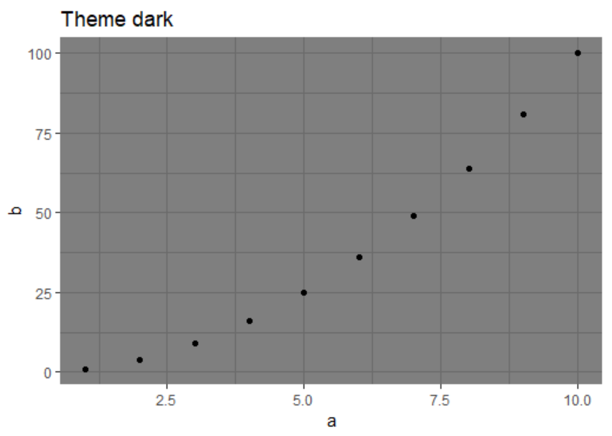
- `theme_grey()`
- `theme_bw()`
- `theme_dark()`
- `theme_light()`
- `theme_minimal()`
- `theme_classic()`
- `theme_linedraw()`

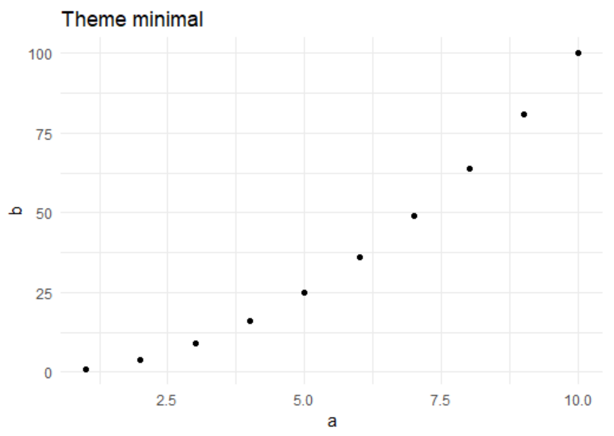












Setting and Customising themes

- Globally
```
theme_set(theme_bw(base_size=14))
```

- In a single plot
```
+theme_dark()
```

Customising themes

```r
theme_update(plot.title = element_text(hjust = 0.5))

plot + theme(plot.title = element_text(hjust = 0.5))
```

```r
theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x, axis.title.x.top, axis.title.x.bottom,
axis.title.y, axis.title.y.left,axis.title.y.right, axis.text, axis.text.x, axis.text.x.top,
axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right, axis.ticks, axis.ticks.x,
axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length,
axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y, axis.line.y.left, axis.line.y.right,
legend.background, legend.margin, legend.spacing, legend.spacing.x, legend.spacing.y, legend.key,
legend.key.size, legend.key.height, legend.key.width, legend.text, legend.text.align, legend.title,
legend.title.align, legend.position, legend.direction, legend.justification, legend.box, legend.box.just,
legend.box.margin, legend.box.background, legend.box.spacing, panel.background, panel.border, panel.spacing,
panel.spacing.x, panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor, panel.grid.major.x,
panel.grid.major.y, panel.grid.minor.x, panel.grid.minor.y, panel.ontop, plot.background, plot.title,
plot.subtitle, plot.caption, plot.tag, plot.tag.position, plot.margin, strip.background, strip.background.x,
strip.background.y, strip.placement, strip.text, strip.text.x, strip.text.y, strip.switch.pad.grid,
strip.switch.pad.wrap)
```

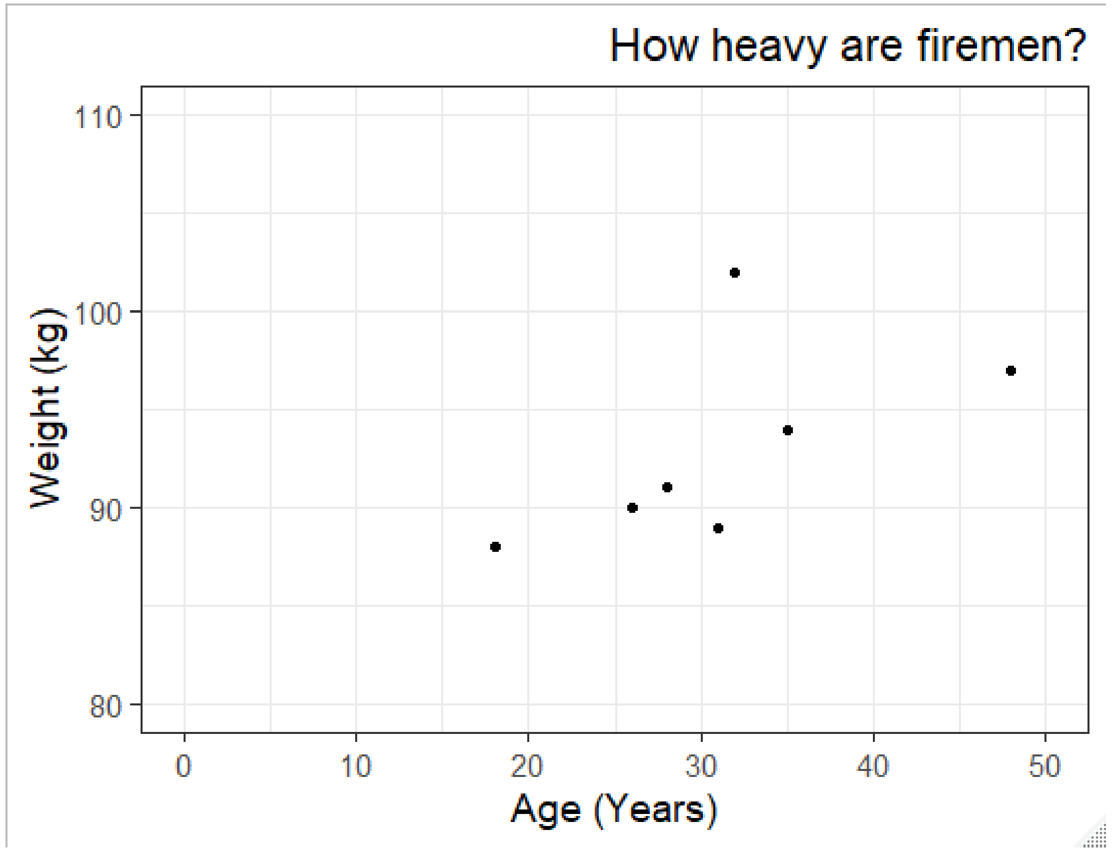
https://ggplot2.tidyverse.org/reference/theme.html

Theme setting example

```
theme_set(theme_bw(base_size = 14))
theme_update(plot.title = element_text(hjust=1))
```

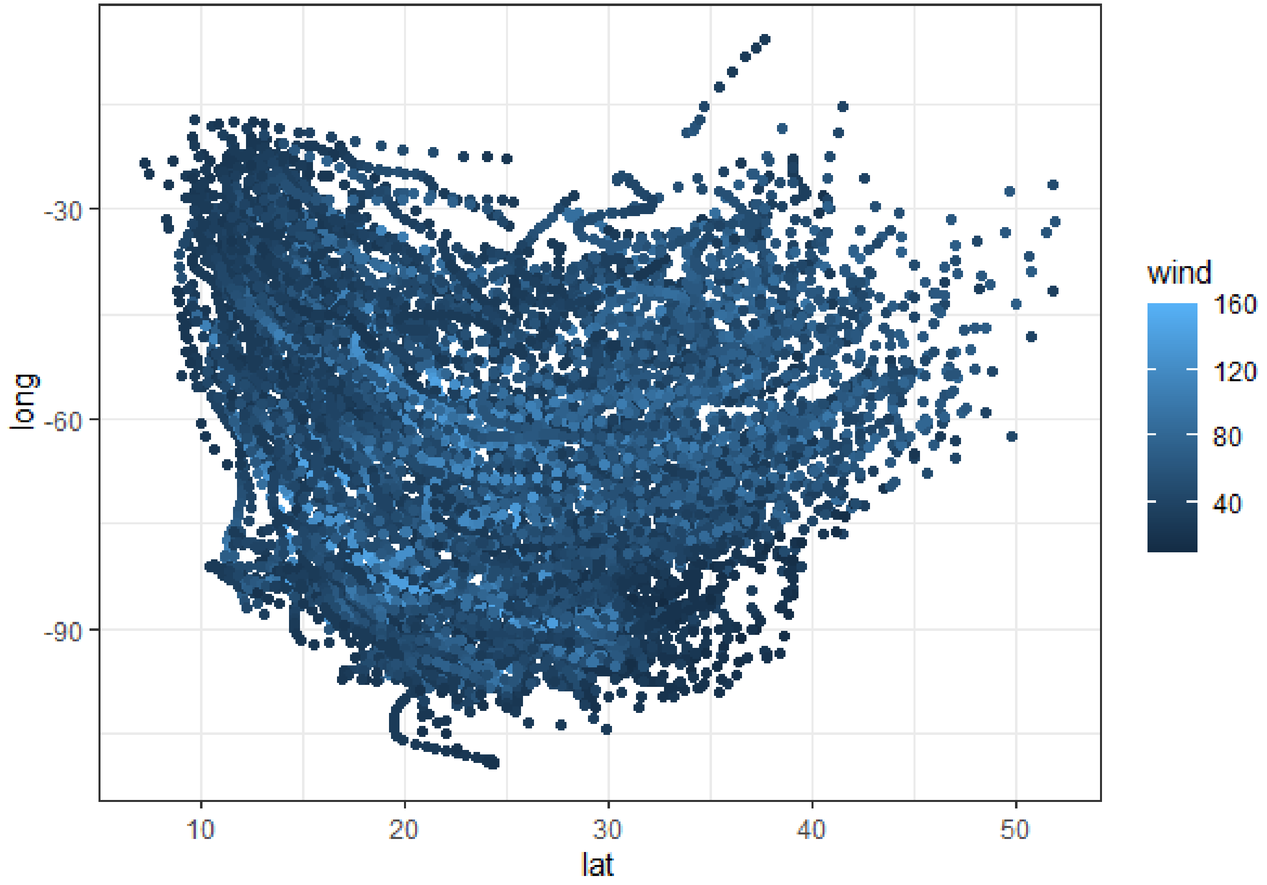
OR

```
my.plot +
theme_bw(base_size = 14) +
theme(plot.title = element_text(hjust=1))
```



Changing Quantitative Colours

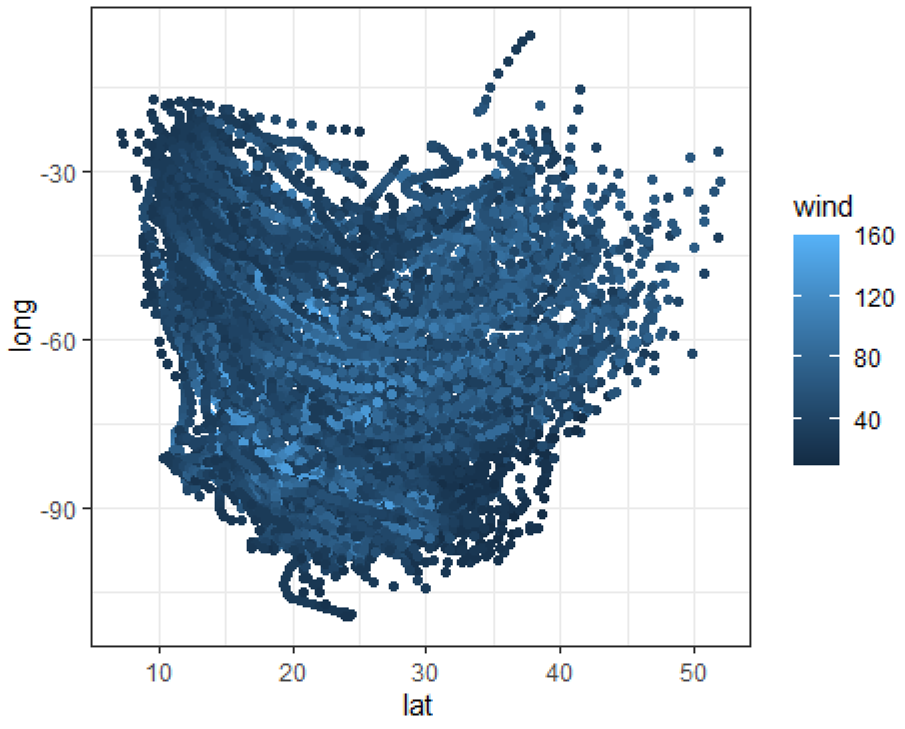
```
storms %>%
  ggplot(aes(x=lat, y=long, colour=wind)) +
  geom_point()
```

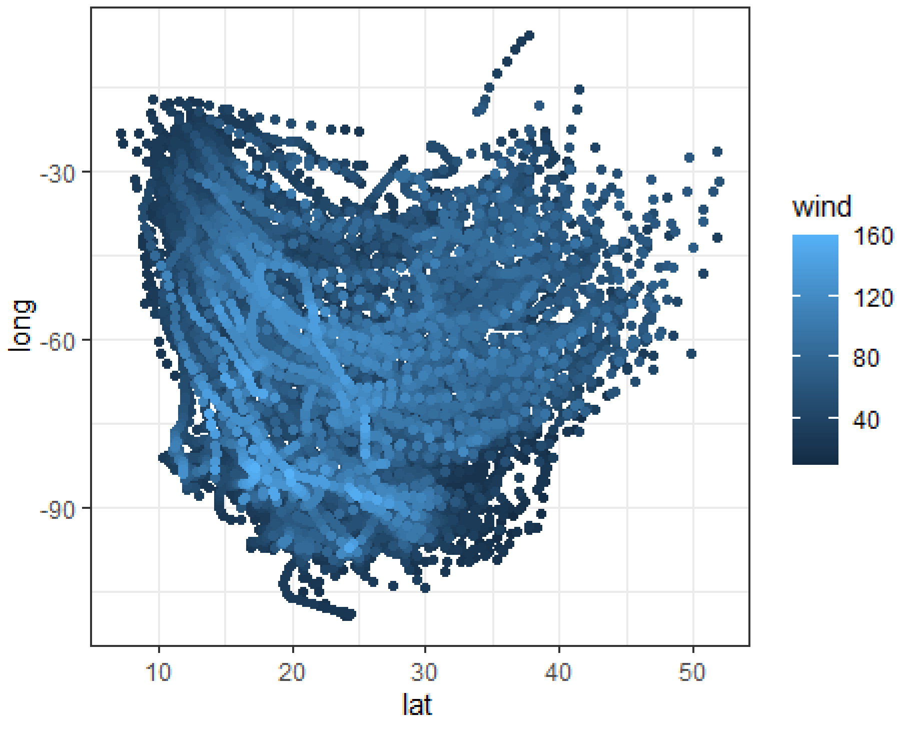


Changing Plotting Order

```
storms %>%
 ggplot(aes(x=lat,y=long,colour=wind))+
 geom_point()
```

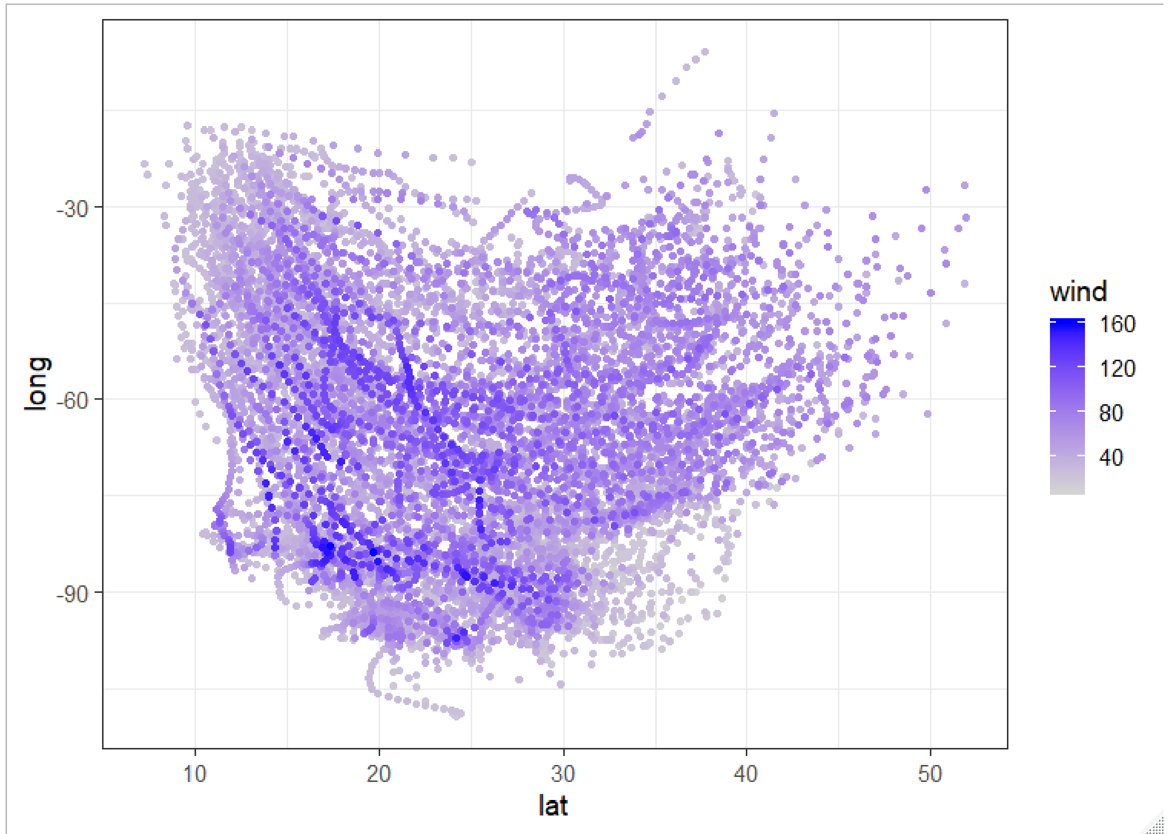
```
storms %>%
 arrange(wind) %>%
 ggplot(aes(x=lat,y=long,colour=wind))+
 geom_point()
```





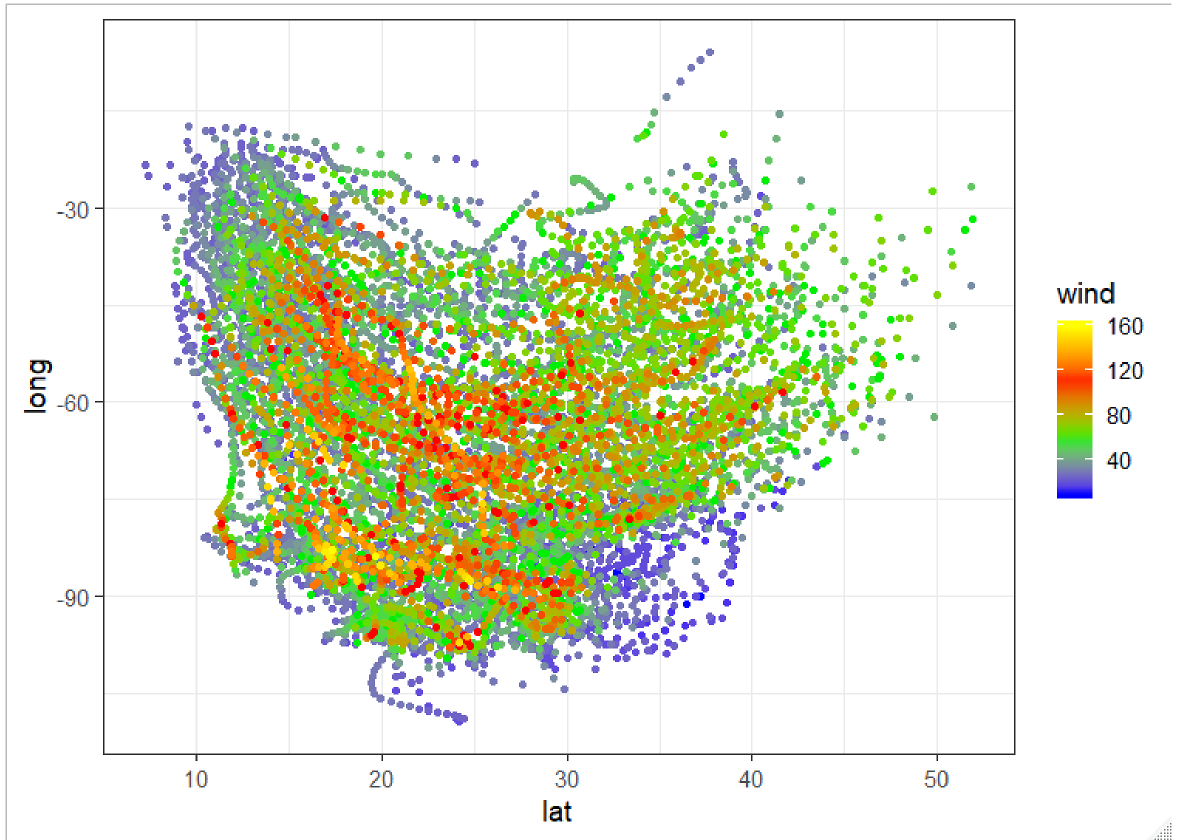
Changing Quantitative Colours

```
storms %>%
  arrange(wind) %>%
  ggplot(aes(x=lat, y=long, colour=wind))+
  geom_point() +
  scale_colour_gradient(low="lightgrey", high="blue")
```

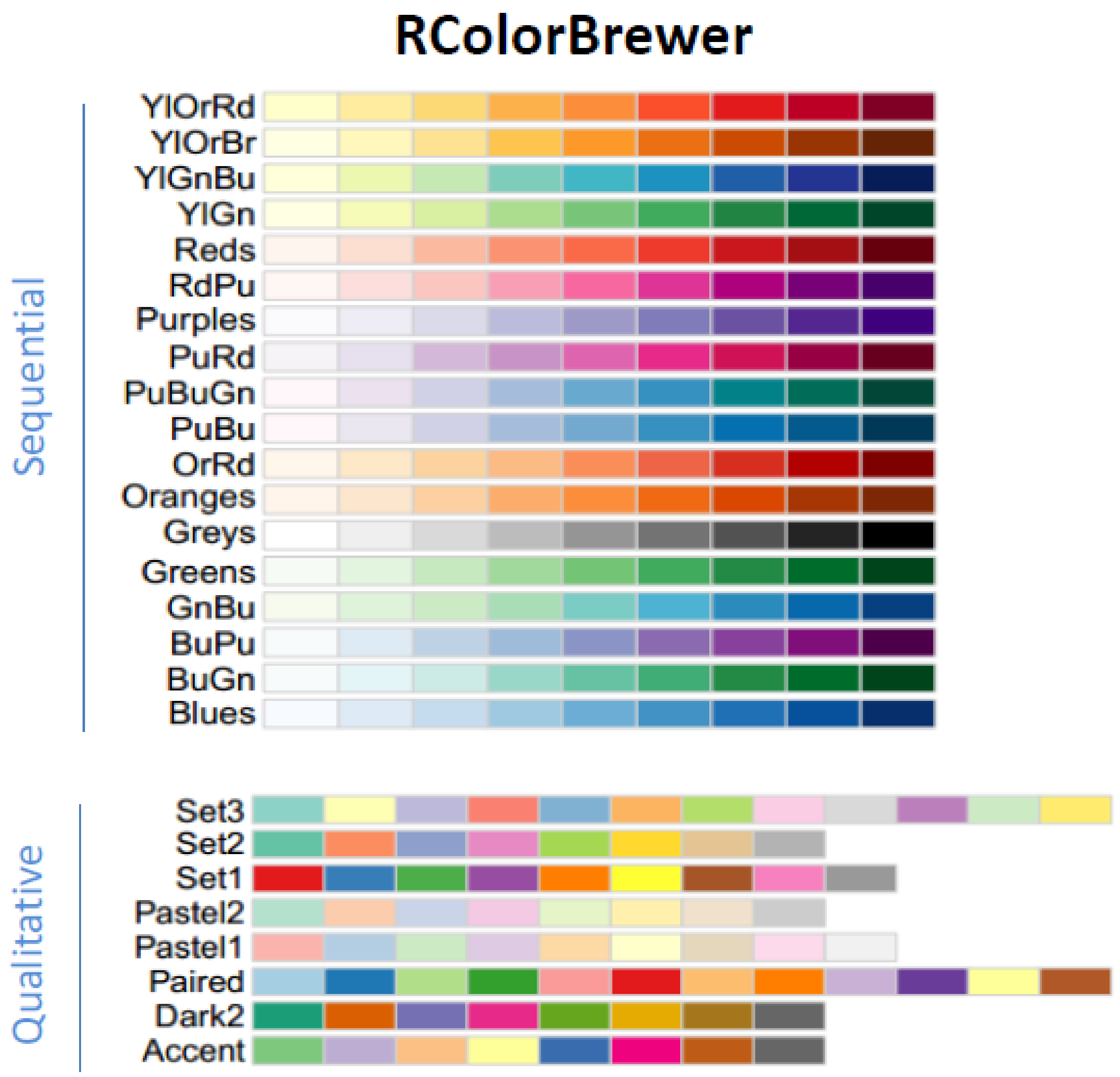


Changing Quantitative Colours

```
storms %>%
  arrange(wind) %>%
  ggplot(aes(x=lat, y=long, colour=wind))+
  geom_point() +
  scale_colour_gradientn(colours=c("blue","green2","red","yellow"))
```



ColorBrewer Scales



Quantitative

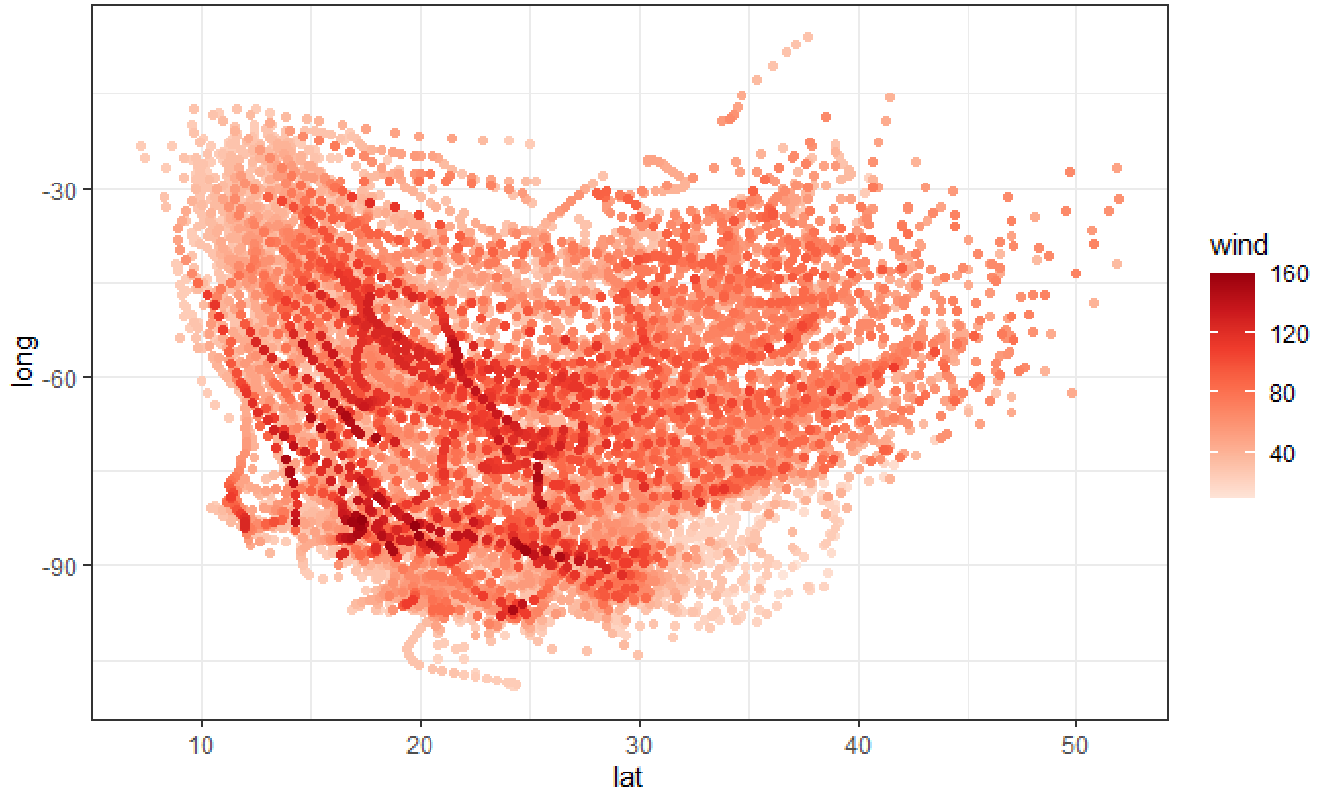
`scale_colour_distiller`

Categorical

`scale_colour_brewer`

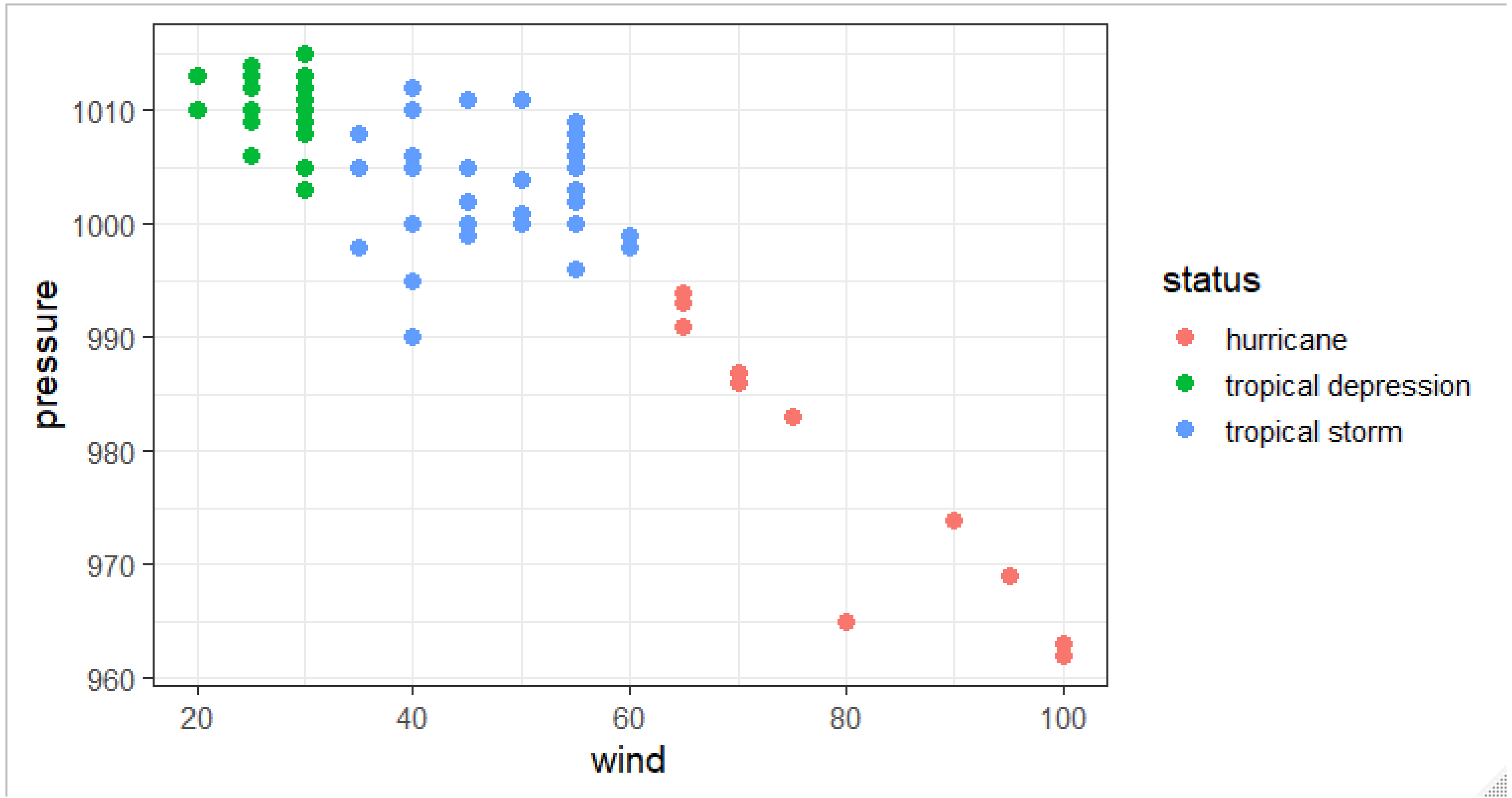
Changing Quantitative Colours

```
storms %>%
  arrange(wind) %>%
  ggplot(aes(x=lat, y=long, color=wind))+
  geom_point() +
  scale_color_distiller(palette="Reds", direction = 1)
```



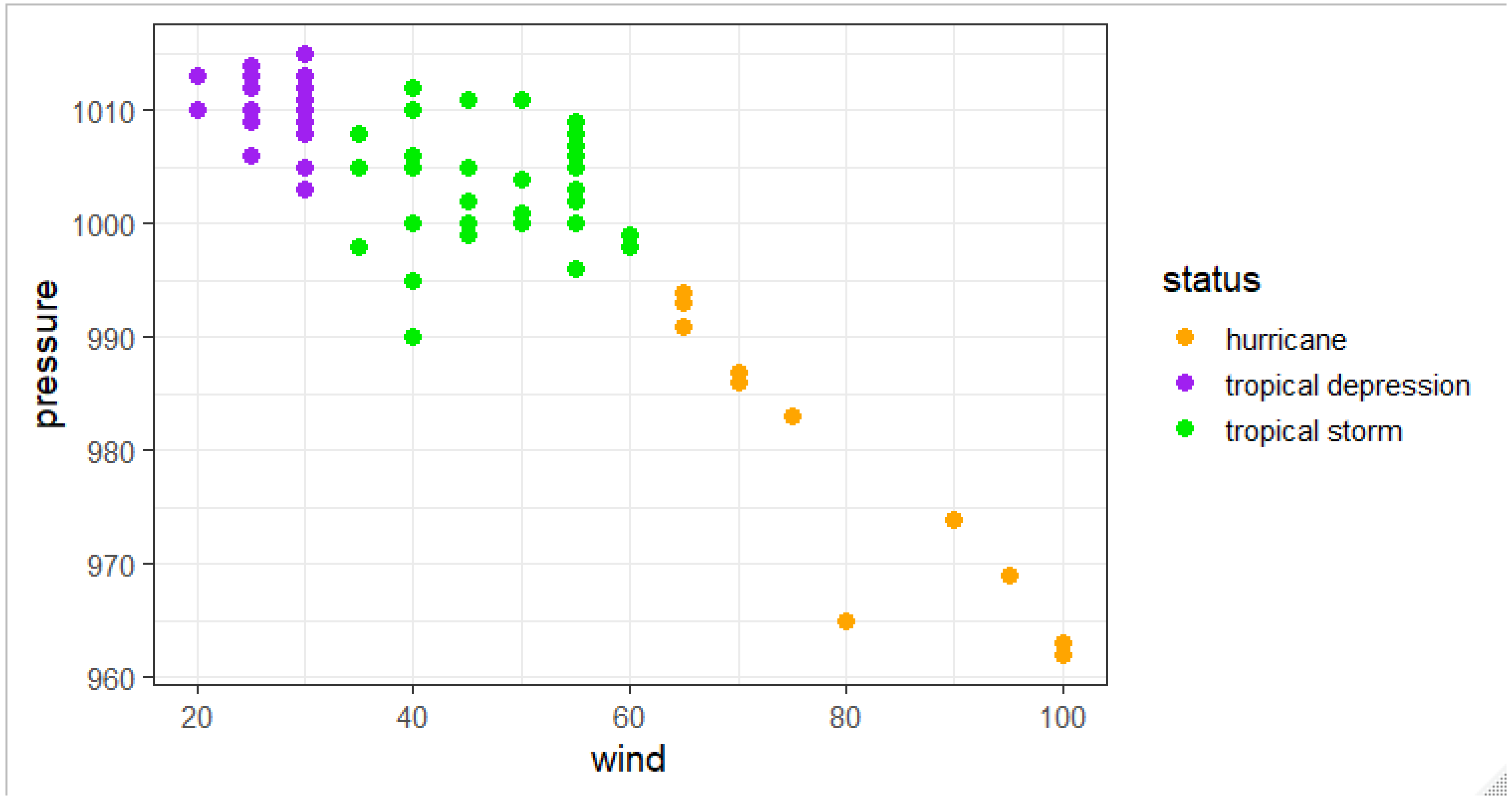
Changing Categorical Colours

```
storms %>%
  filter(year==1983) %>%
  ggplot(aes(x=wind, y=pressure, colour=status)) +
  geom_point(size=3)
```



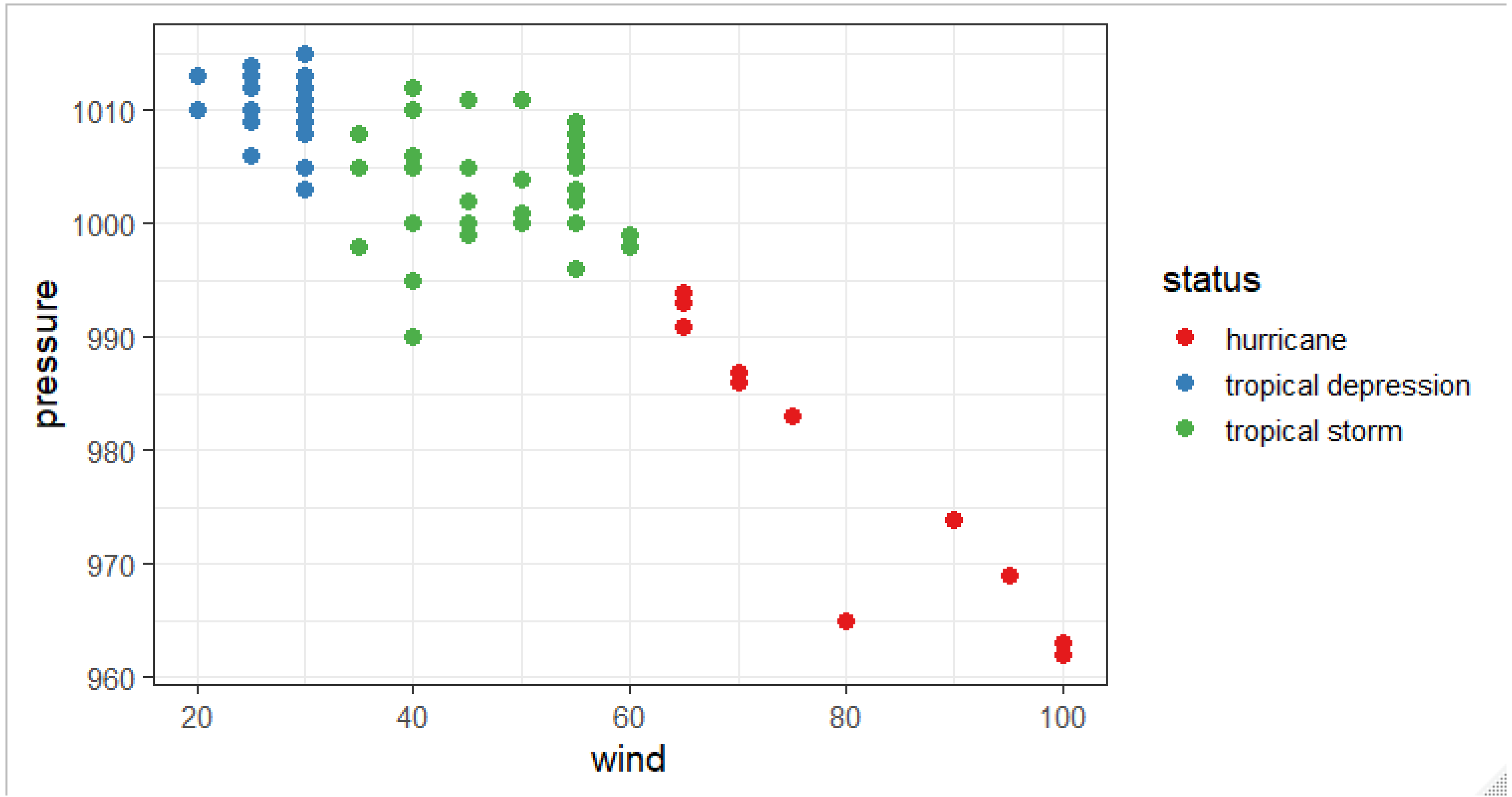
Changing Categorical Colours

```
storms %>%
  filter(year==1983) %>%
  ggplot(aes(x=wind,y=pressure, colour=status)) +
  geom_point(size=3) +
  scale_colour_manual(values = c("orange","purple","green2"))
```



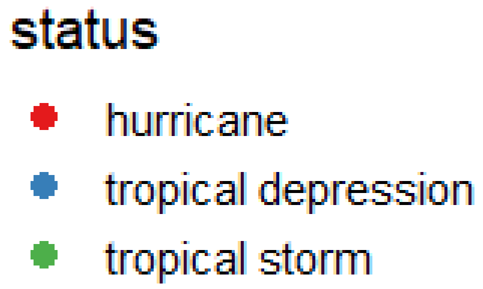
Changing Categorical Colours

```
storms %>%
  filter(year==1983) %>%
  ggplot(aes(x=wind,y=pressure, colour=status)) +
  geom_point(size=3) +
  scale_colour_brewer(palette="Set1")
```



Categorical Colour Ordering

```
# A tibble: 10,010 x 6
     lat  long status              category  wind pressure
   <dbl> <dbl> <chr>               <ord>    <int>    <int>
 1  27.5 -79   tropical depression -1          25     1013
 2  28.5 -79   tropical depression -1          25     1013
 3  29.5 -79   tropical depression -1          25     1013
 4  30.5 -79   tropical depression -1          25     1013
 5  31.5 -78.8 tropical depression -1          25     1012
 6  32.4 -78.7 tropical depression -1          25     1012
 7  33.3 -78   tropical depression -1          25     1011
 8  34   -77   tropical depression -1          30     1006
 9  34.4 -75.8 tropical storm      0           35     1004
10  34   -74.8 tropical storm      0           40     1002
# ... with 10,000 more rows
```



Status is a character vector – ordering is alphabetical

Factors

- Similar to text (character) vectors, but with some differences
  - They have controlled values – you can limit which values can be added
  - The values which can go in are tracked separately to the data
  - The values which can go in have an explicit order

- GGplot respects the ordering of factors, so converting to factors is the simplest way to re-order a plot

Converting character vectors to factors

```
> chr.names
 [1] "simon" "anne"  "laura" "felix" "simon" "anne"  "laura"
 [8] "felix" "simon" "anne"  "laura" "felix" "simon" "anne"
[15] "laura" "felix" "simon" "anne"  "laura" "felix"


> factor(chr.names)
 [1] simon anne  laura felix simon anne  laura felix simon
[10] anne  laura felix simon anne  laura felix simon anne
[19] laura felix
Levels: anne felix laura simon


> factor(chr.names, levels=c("simon","anne","laura","felix"))
 [1] simon anne  laura felix simon anne  laura felix simon
[10] anne  laura felix simon anne  laura felix simon anne
[19] laura felix
Levels: simon anne laura felix
```

Categorical Colour Ordering

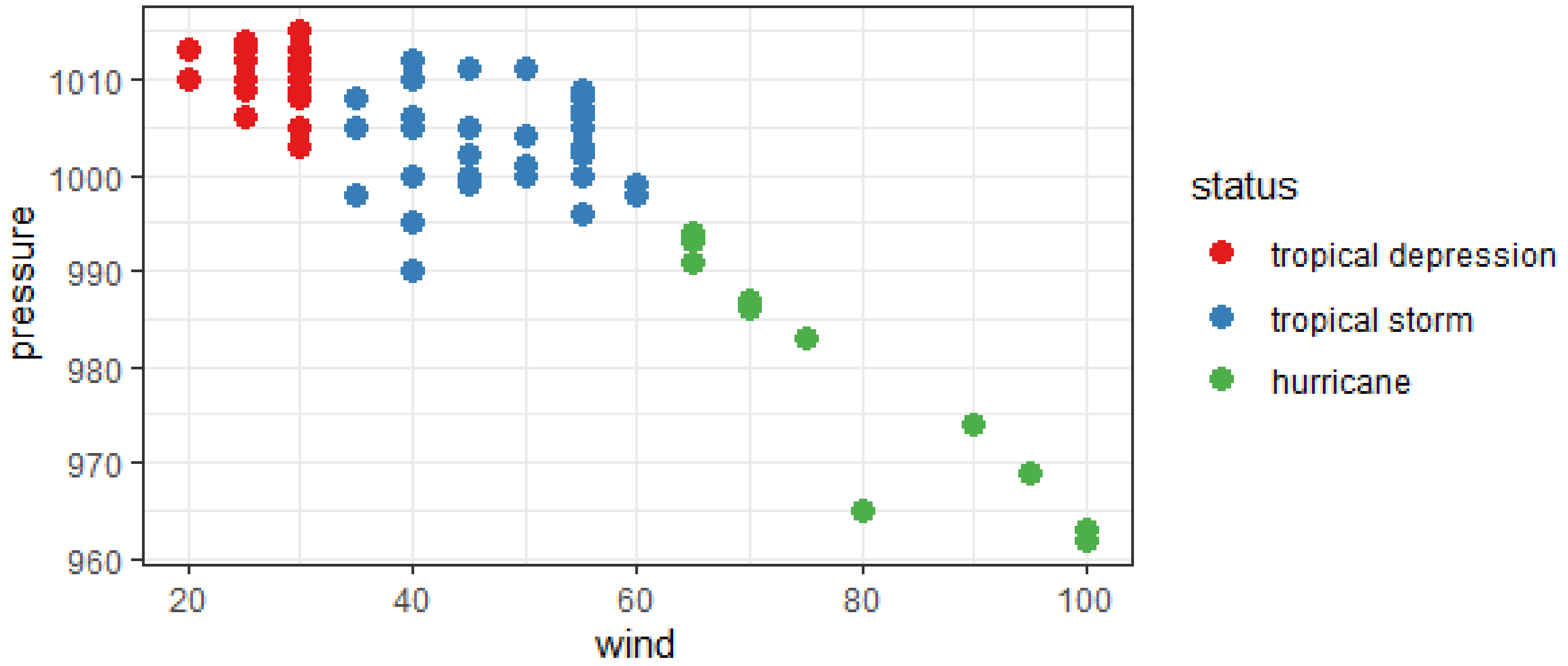
Use factors for explicit ordering

```
storms %>%
  mutate(
    status=factor(
            status,
            levels=c("tropical depression","tropical storm","hurricane")
            )
  )
```

```
# A tibble: 10,010 x 6
     lat  long status                 category  wind pressure
   <dbl> <dbl> <fct>                  <ord>    <int>    <int>
 1  27.5 -79   tropical depression -1             25     1013
 2  28.5 -79   tropical depression -1             25     1013
 3  29.5 -79   tropical depression -1             25     1013
 4  30.5 -79   tropical depression -1             25     1013
```

Categorical Colour Ordering

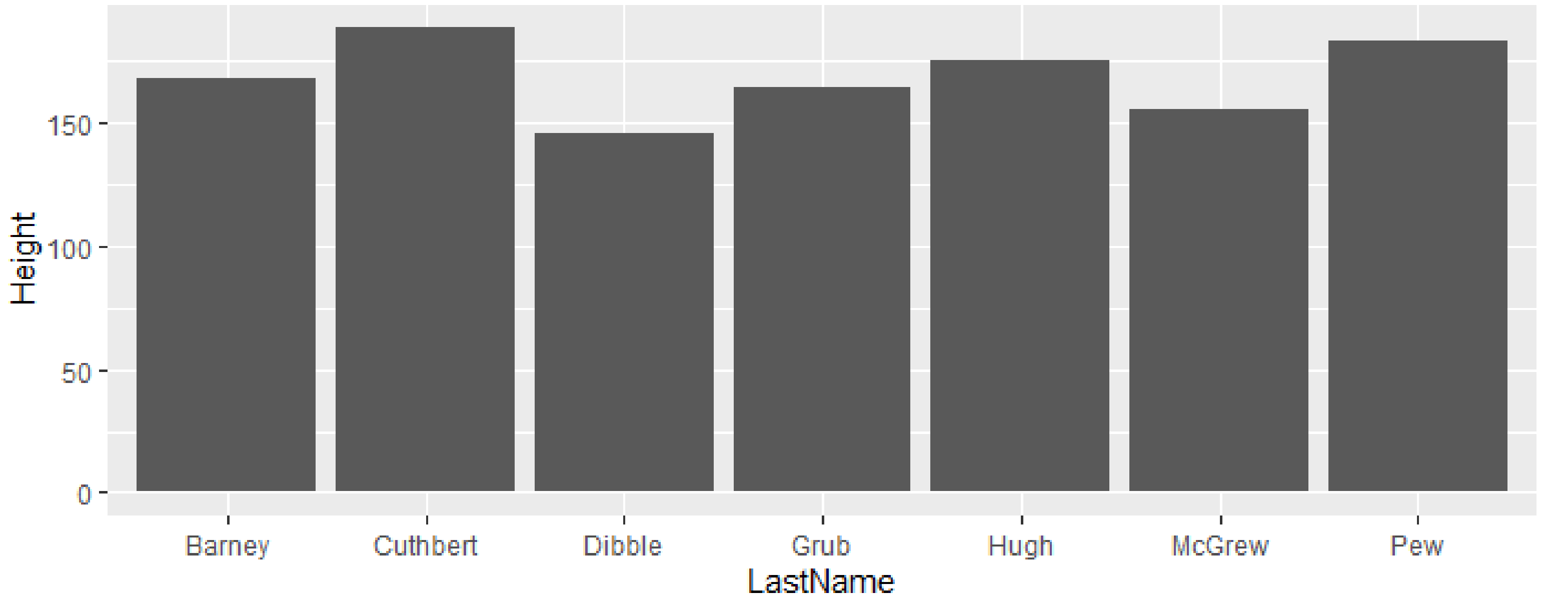
```
storms %>%
  mutate(status=factor(status, levels=c("tropical depression","tropical storm","hurricane"))) %>%
  filter(year==1983) %>%
  ggplot(aes(x=wind,y=pressure, colour=status)) +
  geom_point(size=3)+
  scale_color_brewer(palette="Set1")
```



```
  LastName FirstName   Age Weight Height
  <chr>    <chr>     <dbl>  <dbl>  <dbl>
1 Hugh     Chris        26     90    175
2 Pew      Adam         32    102    183
3 Barney   Daniel       18     88    168
4 McGrew   Chris        48     97    155
5 Cuthbert Carl         28     91    188
6 Dibble   Liam         35     94    145
7 Grub     Doug         31     89    164
```

Reordering example
Keep the original order

```
trumpton %>%
  ggplot(aes(x=LastName, y=Height)) +
  geom_col()
```

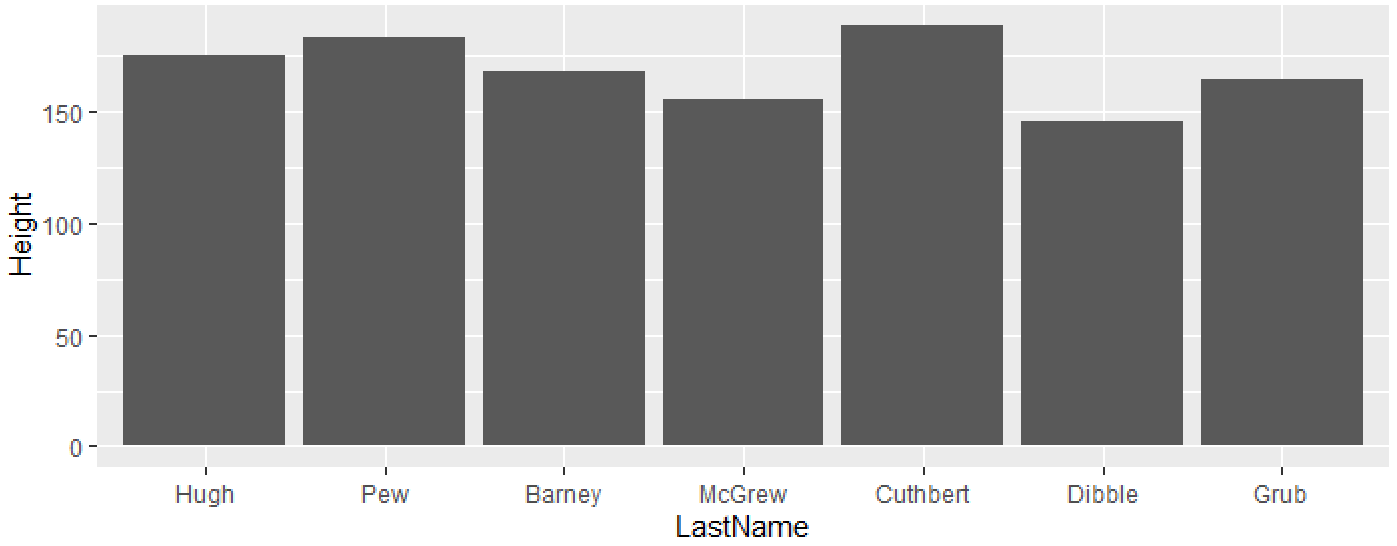


The default is to order alphabetically

Reordering example
Keep the original order

```
  LastName FirstName   Age Weight Height
  <chr>    <chr>     <dbl>  <dbl>  <dbl>
1 Hugh     Chris        26     90    175
2 Pew      Adam         32    102    183
3 Barney   Daniel       18     88    168
4 McGrew   Chris        48     97    155
5 Cuthbert Carl         28     91    188
6 Dibble   Liam         35     94    145
7 Grub     Doug         31     89    164
```

```
trumpton %>%
  mutate(LastName=factor(LastName, levels=LastName)) %>%
  ggplot(aes(x=LastName, y=Height)) +
  geom_col()
```



We can convert to a factor and use `levels` to enforce the same order. If we had just converted to a factor it would have been alphabetical still.

Quantitative ordering with reorder

- The reorder function allows you to order the levels of a factor by a different quantitative variable

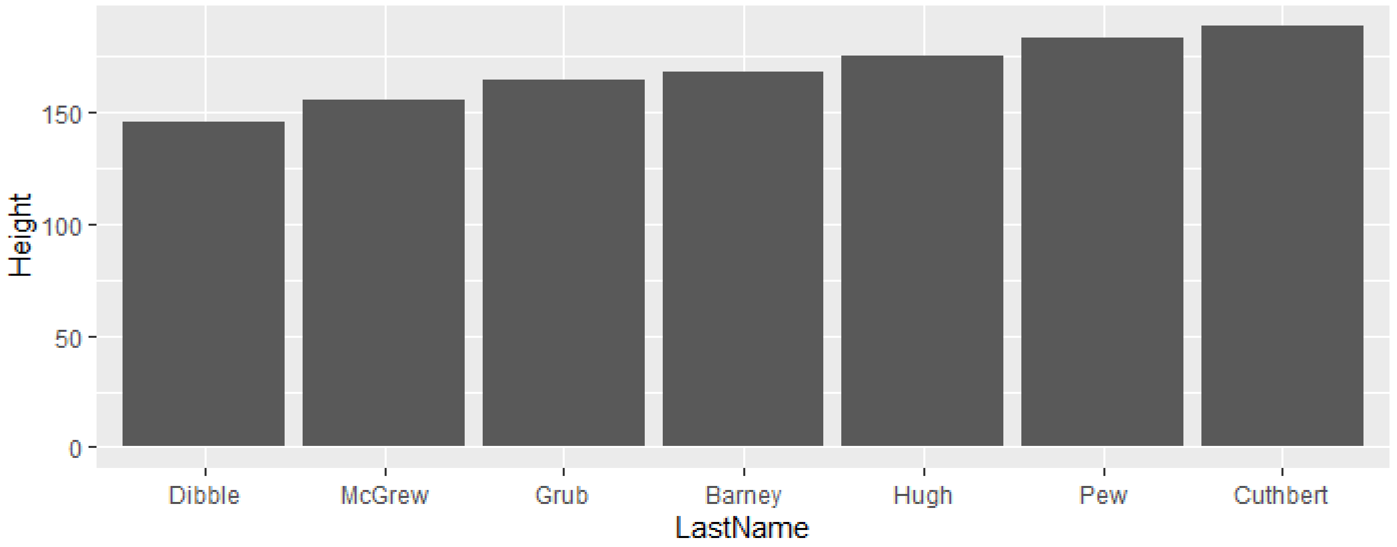
- It allows you to sort a figure by value

- `reorder(categorical, quantitative)`

Reordering examples

```
  LastName FirstName   Age Weight Height
  <chr>    <chr>     <dbl>  <dbl>  <dbl>
1 Hugh     Chris        26     90    175
2 Pew      Adam         32    102    183
3 Barney   Daniel       18     88    168
4 McGrew   Chris        48     97    155
5 Cuthbert Carl         28     91    188
6 Dibble   Liam         35     94    145
7 Grub     Doug         31     89    164
```

```
trumpton %>%
  mutate(LastName=reorder(LastName,Height)) %>%
  ggplot(aes(x=LastName, y=Height)) +
  geom_col()
```

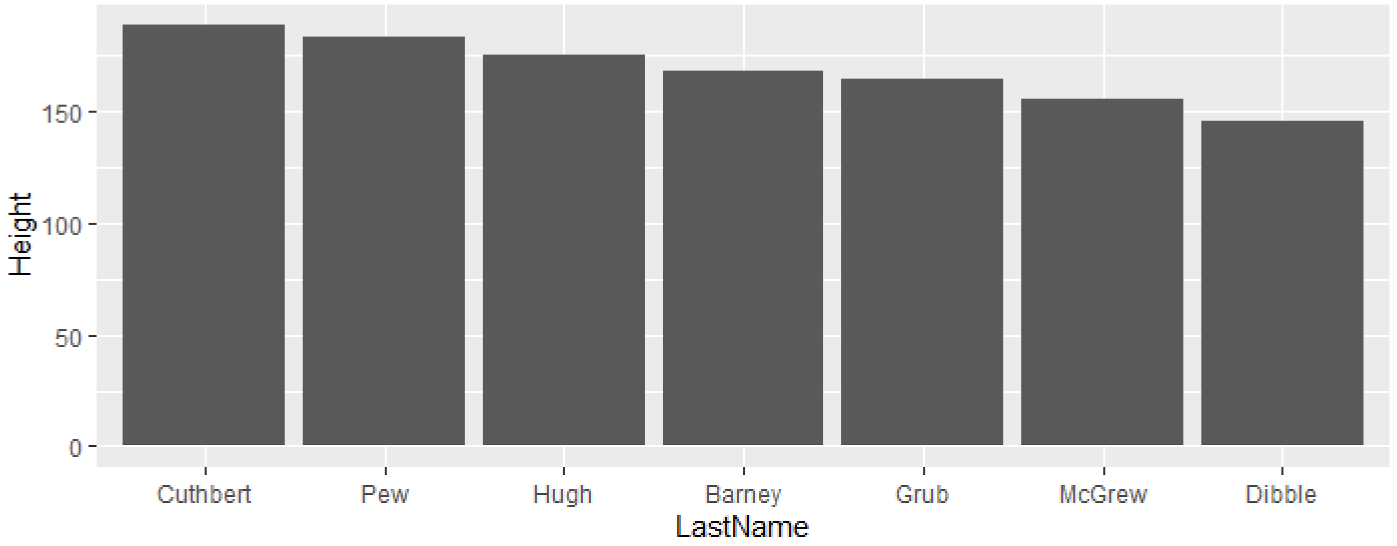


By using `reorder` we can make the levels correspond to a quantitative variable. Here it is the same one we're plotting, but it doesn't have to be.

```
  LastName FirstName   Age Weight Height
  <chr>    <chr>     <dbl>  <dbl>  <dbl>
1 Hugh     Chris        26     90    175
2 Pew      Adam         32    102    183
3 Barney   Daniel       18     88    168
4 McGrew   Chris        48     97    155
5 Cuthbert Carl         28     91    188
6 Dibble   Liam         35     94    145
7 Grub     Doug         31     89    164
```

Reordering examples

```
trumpton %>%
  mutate(LastName=reorder(LastName,-Height)) %>%
  ggplot(aes(x=LastName, y=Height)) +
  geom_col()
```



We can use `-Height` in the reorder to reverse the sorting order

Exercise 3

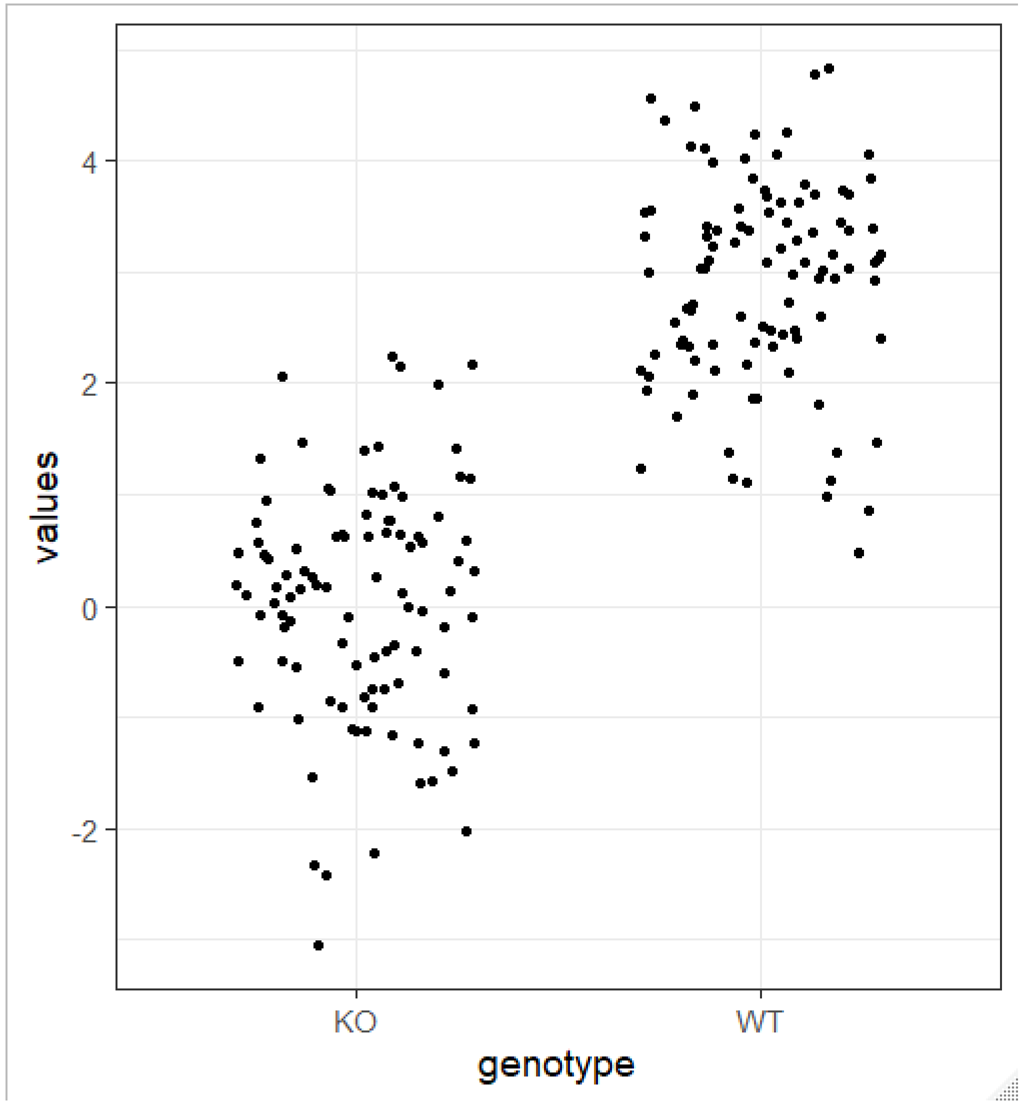
Babraham Bioinformatics

Statistical Overlays

Babraham Bioinformatics

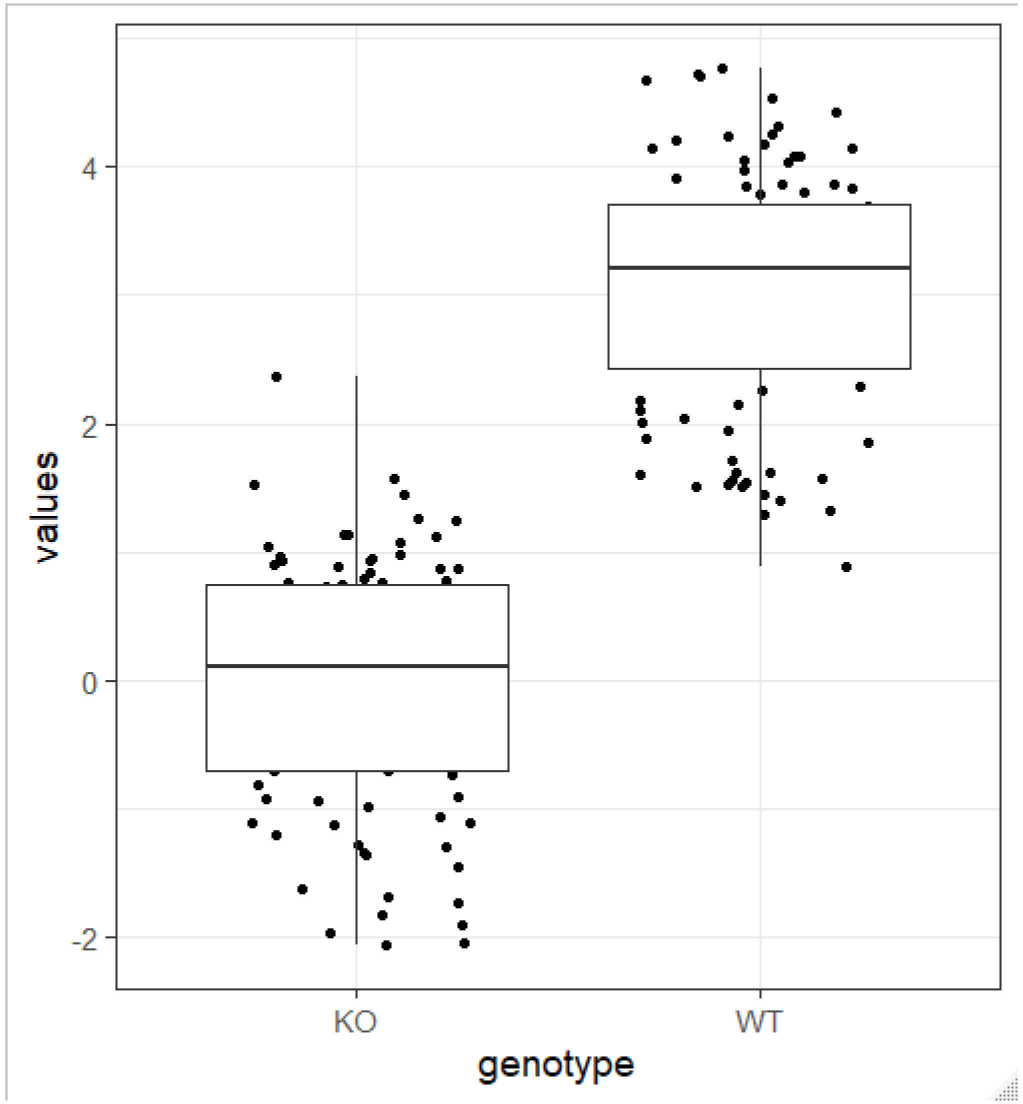
Overlaying raw data and summaries

```
many.values %>%
  group_by(genotype) %>%
  sample_n(100) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_jitter(height=0, width = 0.3)
```



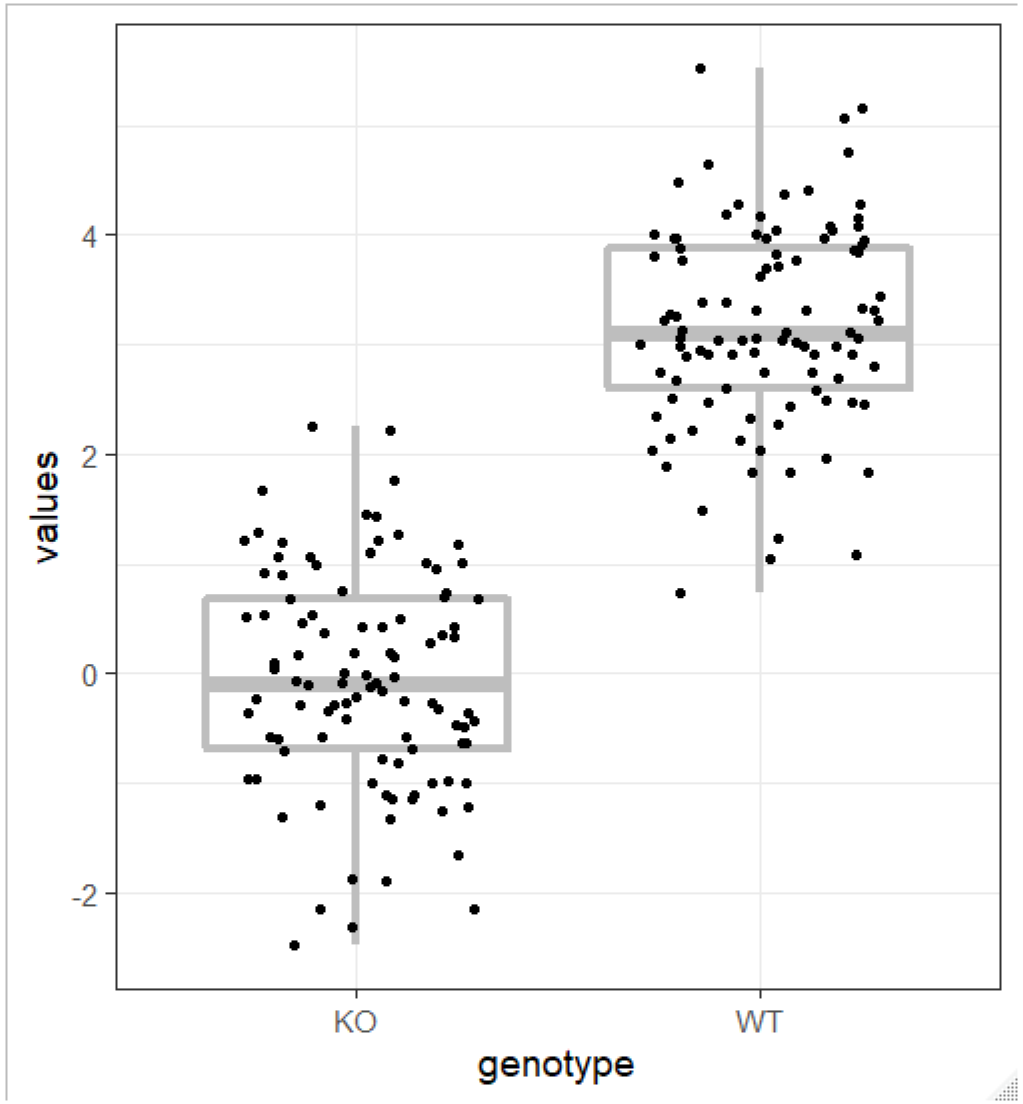
Overlaying raw data and summaries

```
many.values %>%
  group_by(genotype) %>%
  sample_n(100) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_jitter(height=0, width = 0.3) +
  geom_boxplot()
```



Overlaying raw data and summaries

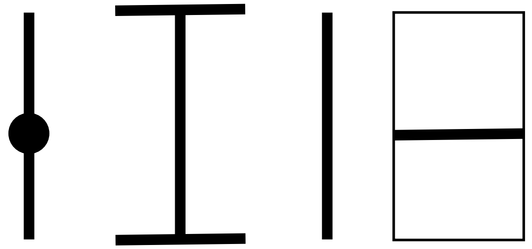
```
many.values %>%
  group_by(genotype) %>%
  sample_n(100) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_boxplot(size=1.5, colour="grey") +
  geom_jitter(height=0, width = 0.3)
```



Stat Summary

- Add summary statistics to discrete data

- Main options
  - `geom` – how is this going to be displayed
    - pointrange (default)
    - errorbar
    - linerange
    - Crossbar

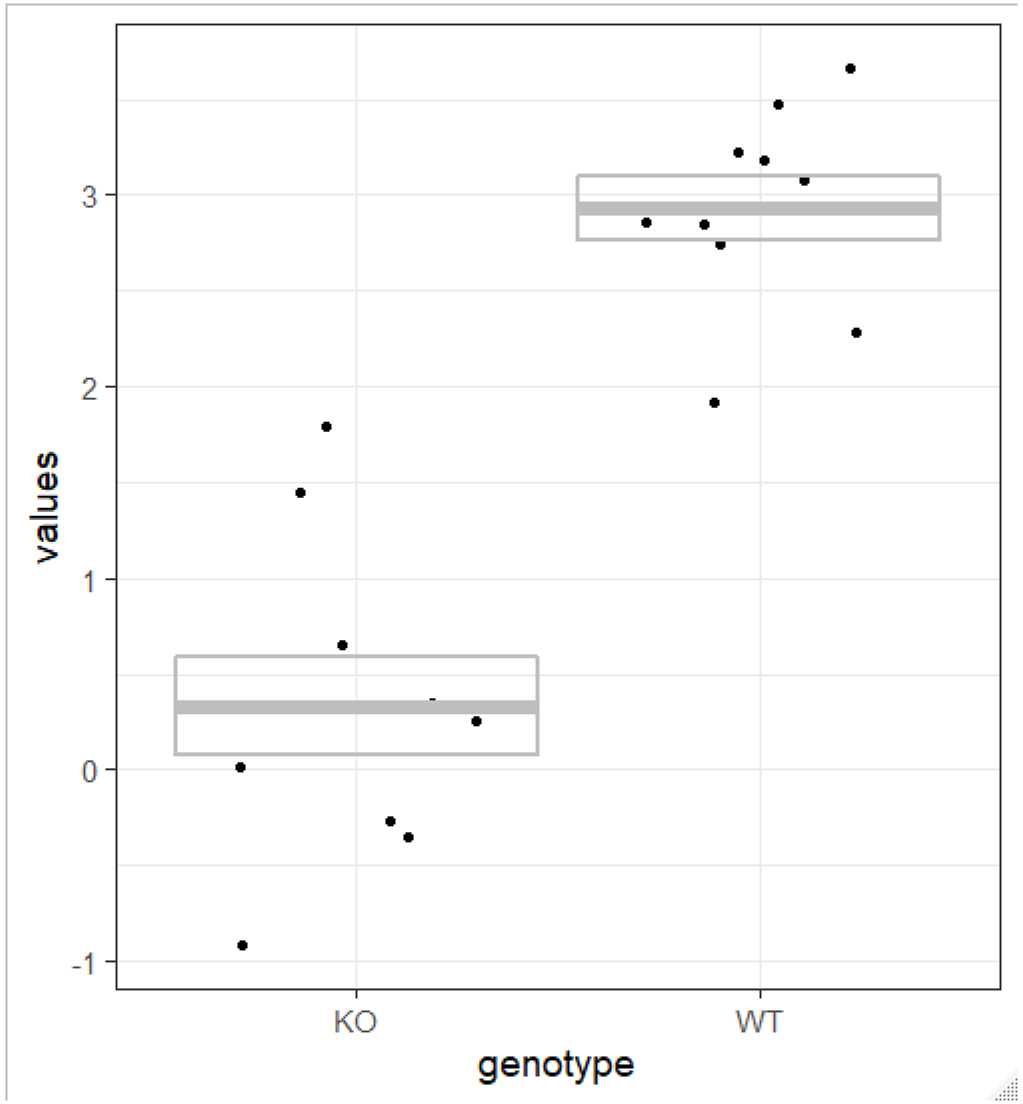


  - `fun.data`
    - Function to produce
      - Min, Centre, Max
      - Eg `mean_se, mean_cl_boot, mean_cl_normal,mean_sdl`

    - Can also use `fun.min`, `fun`, `fun.max` separately

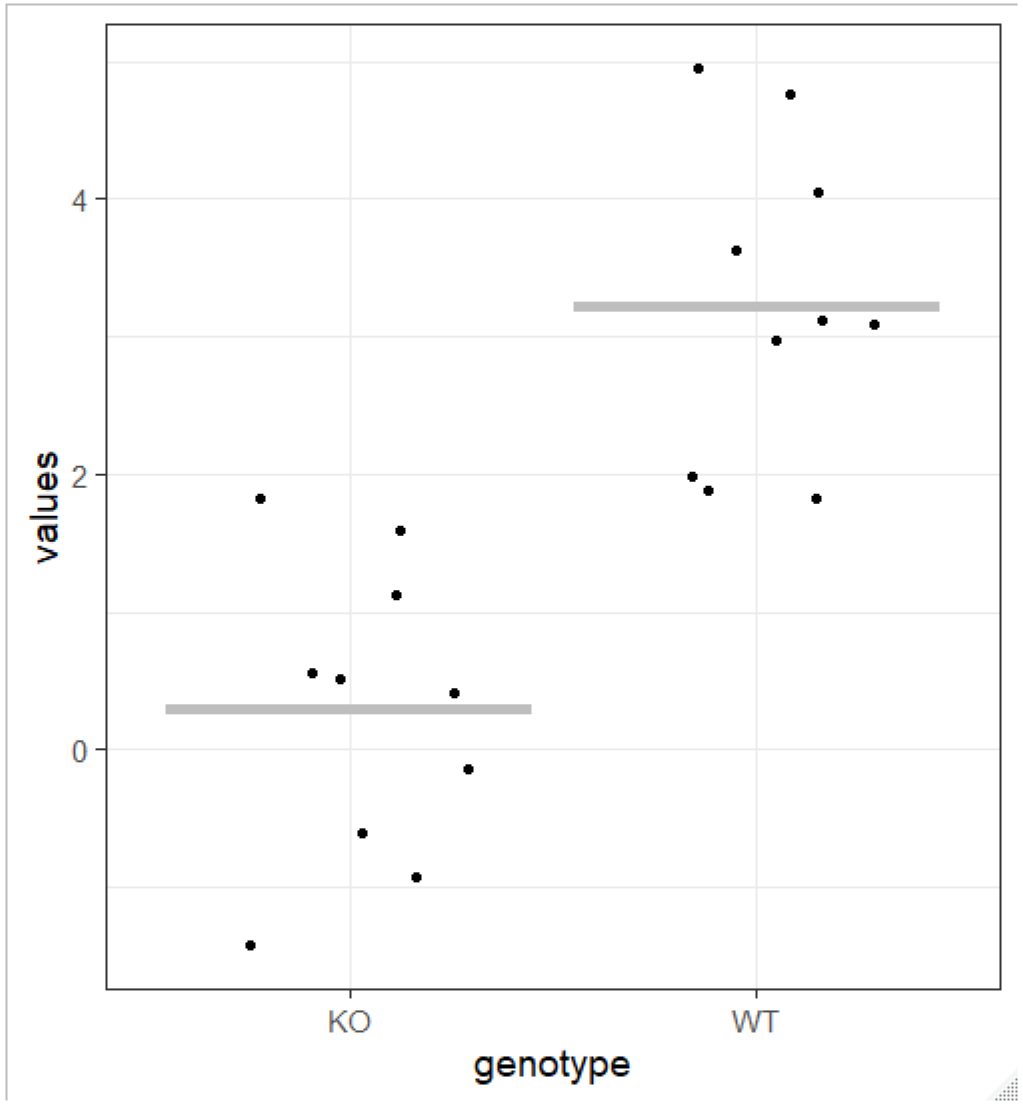
Overlaying raw data and summaries

```
many.values %>%
  group_by(genotype) %>%
  sample_n(10) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_jitter(height=0, width = 0.3) +
  stat_summary(
    geom="crossbar",
    fun.data=mean_se,
    size=1, alpha=0, colour="grey"
  )
```



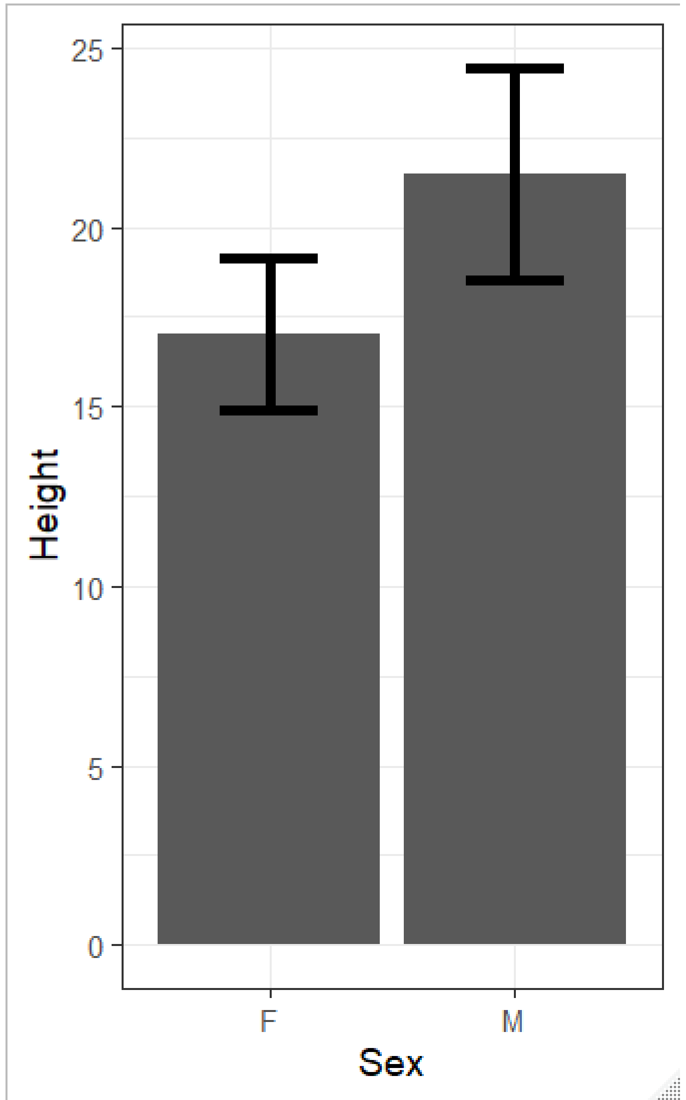
Overlaying raw data and summaries

```
many.values %>%
  group_by(genotype) %>%
  sample_n(10) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_jitter(height=0, width = 0.3) +
  stat_summary(
    geom="errorbar",
    fun     = mean,
    fun.max = mean,
    fun.min = mean,
    size=2,
    colour="grey"
  )
```



Overlaying raw data and summaries

```
group.data %>%
  ggplot(aes(x=Sex, y=Height)) +
  geom_bar(stat="summary", fun=mean) +
  stat_summary(geom="errorbar", width=0.4, size=2)
```

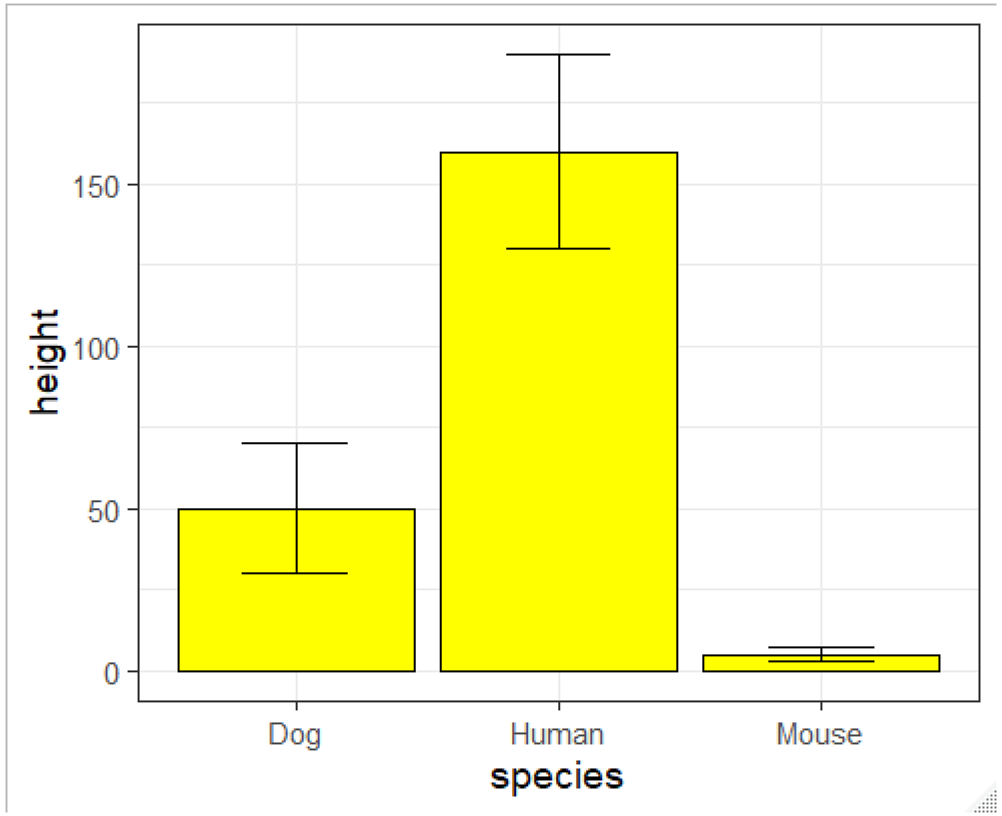


NB The `fun=mean` in `geom_bar` is optional since that's the default

Using pre-calculated variance measures

```
data.with.stdev %>%
  ggplot(aes(x=species,y=height, ymin=height-stdev, ymax=height+stdev)) +
  geom_col(fill="yellow", color="black") +
  geom_errorbar(width=0.4)
```

```
> data.with.stdev
# A tibble: 3 x 3
  species height stdev
  <chr>    <dbl> <dbl>
1 Human      160    30
2 Dog         50    20
3 Mouse        5     2
```



Adding Reference / Regression Lines

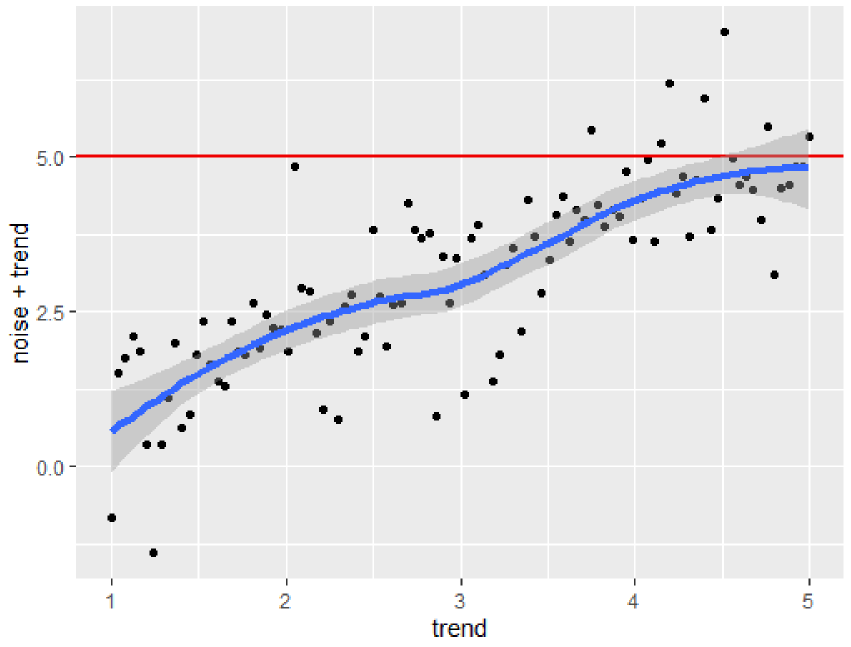
- `geom_hline` – Adds a horizontal line (specify `yintercept`)
- `geom_vline` – Adds a vertical line (specify `xintercept`)

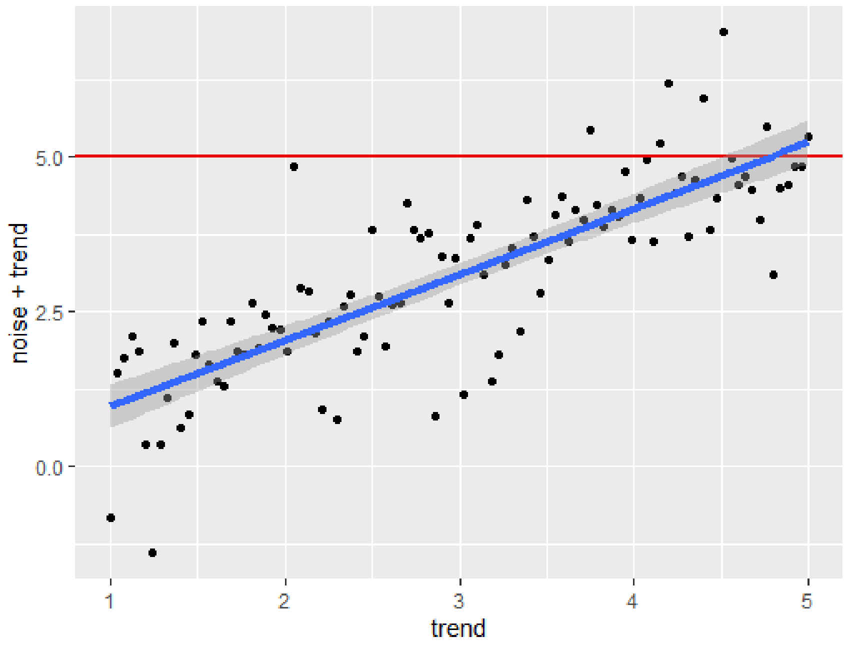
- `geom_abline` – Adds an angled line (specify slope and intercept)
  - Values can come from the `lm` function to generate a linear model
- `geom_smooth` – Calculates and draws regression lines
  - Loess smoothed curves
  - Linear modelled lines

Trend lines

```
trend_data %>%
  ggplot(aes(x=trend,y=noise+trend)) +
  geom_point() +
  geom_hline(
   yintercept=5, size=1, colour="red2") +
  geom_smooth(size=1.5)
```

```
trend_data %>%
  ggplot(aes(x=trend,y=noise+trend)) +
  geom_point() +
  geom_hline(
   yintercept=5, size=1, colour="red2") +
  geom_smooth(size=1.5, method="lm")
```





Exercise 4

Faceting and Highlighting

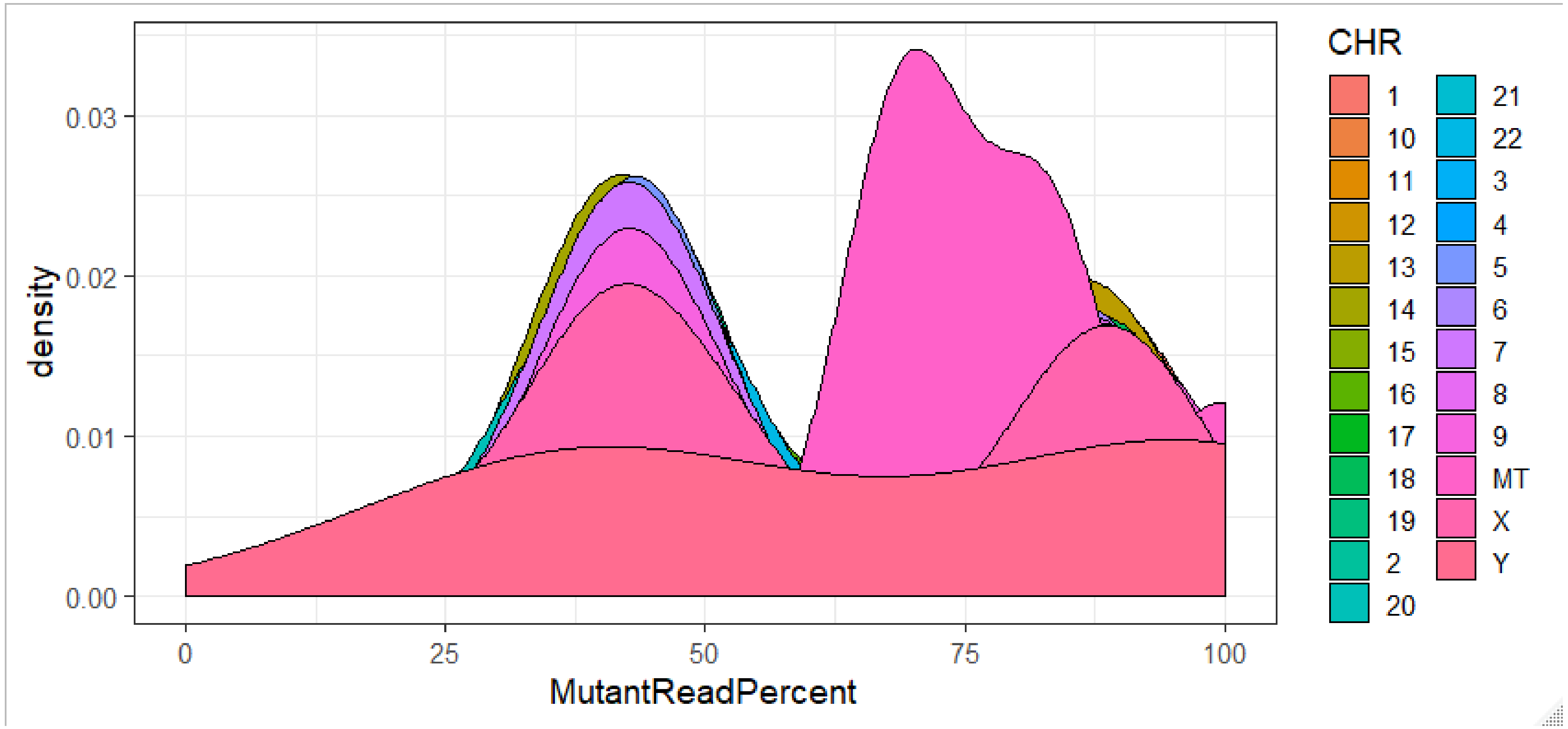
Babraham Bioinformatics

Faceting

- Faceting allows you to take a single graph definition and create multiple graphs of the same type based on additional categorical factors
- `facet_grid` draws graphs in rows and columns based on 1 or 2 factors
- `facet_wrap` draws a 2D arrangement of graphs based on a single factor

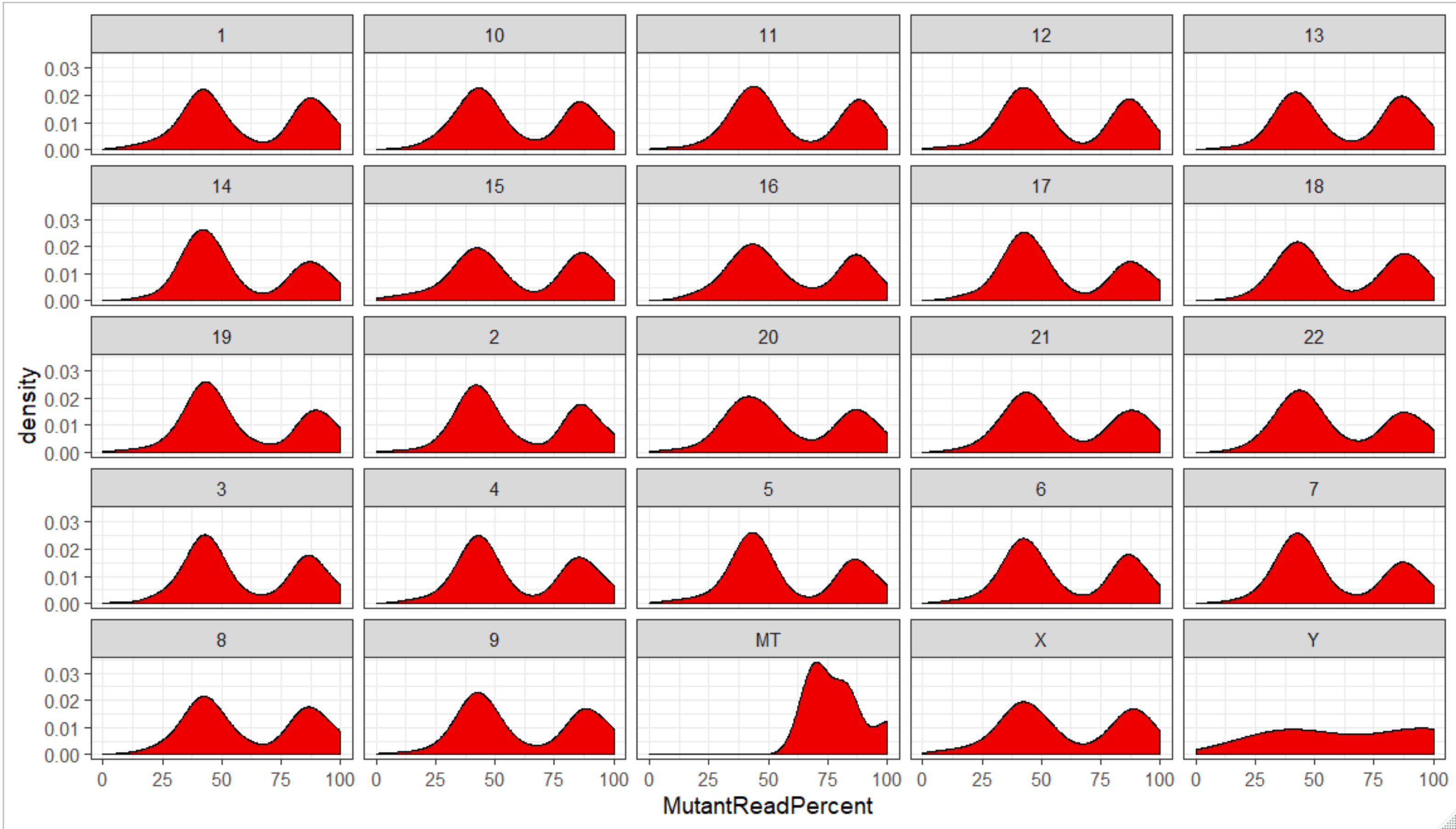
Faceting – using `facet_wrap()`

```
child.variants %>%
  ggplot(aes(x=MutantReadPercent, fill=CHR)) +
  geom_density()
```



Faceting – using `facet_wrap()`

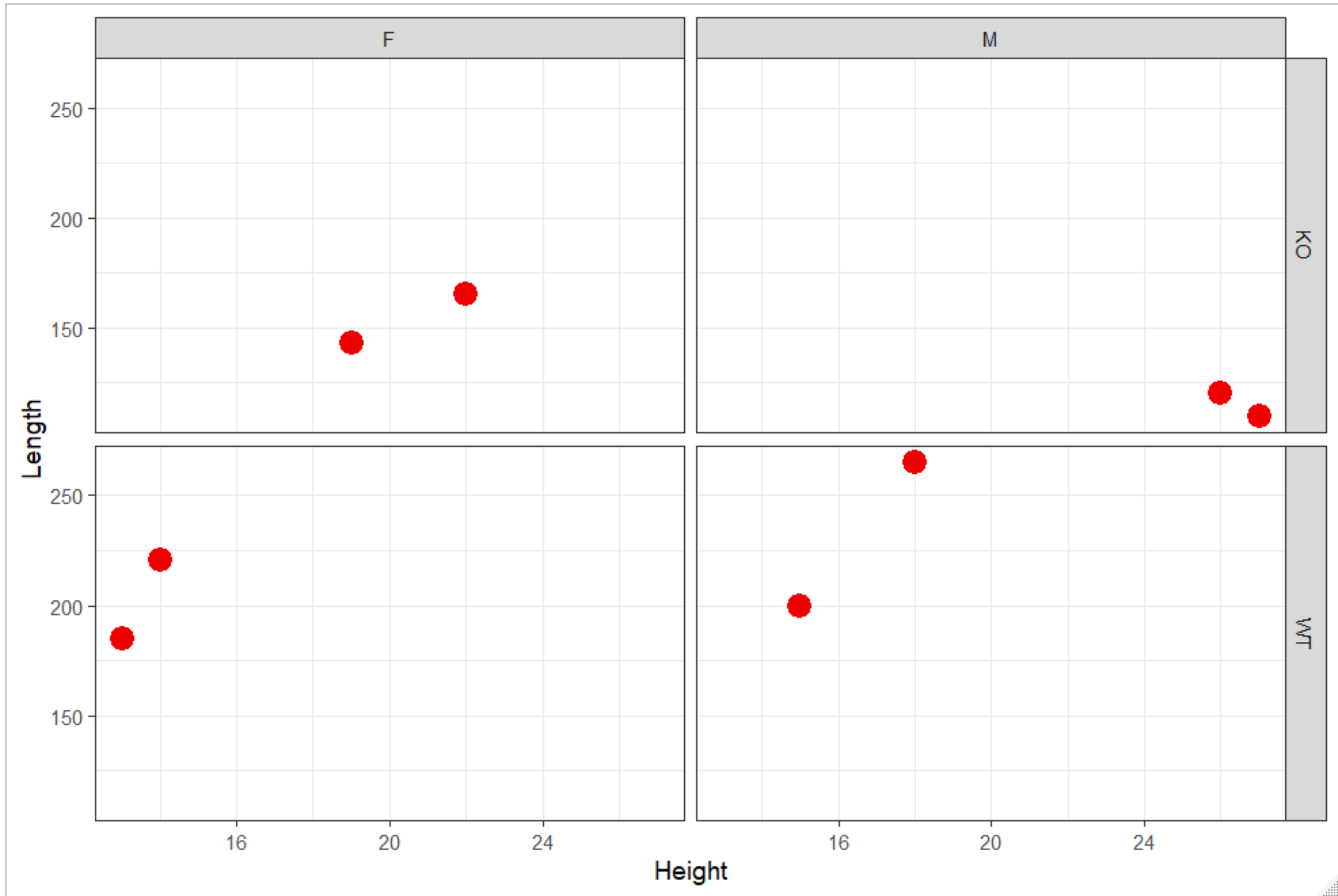
```
child.variants %>%
  ggplot(aes(x=MutantReadPercent)) +
  geom_density(fill="red2") +
  facet_wrap(vars(CHR))
```



Note that the variable defining the facets must be passed through the `vars()` function

Faceting – using `facet_grid()`

```
group.data %>%
  ggplot(aes(x=Height, y=Length)) +
  geom_point(size=6, color="red2") +
  facet_grid(
    rows=vars(Genotype),
    cols=vars(Sex)
  )
```



Note that the variable defining the facets must be passed through the `vars()` function

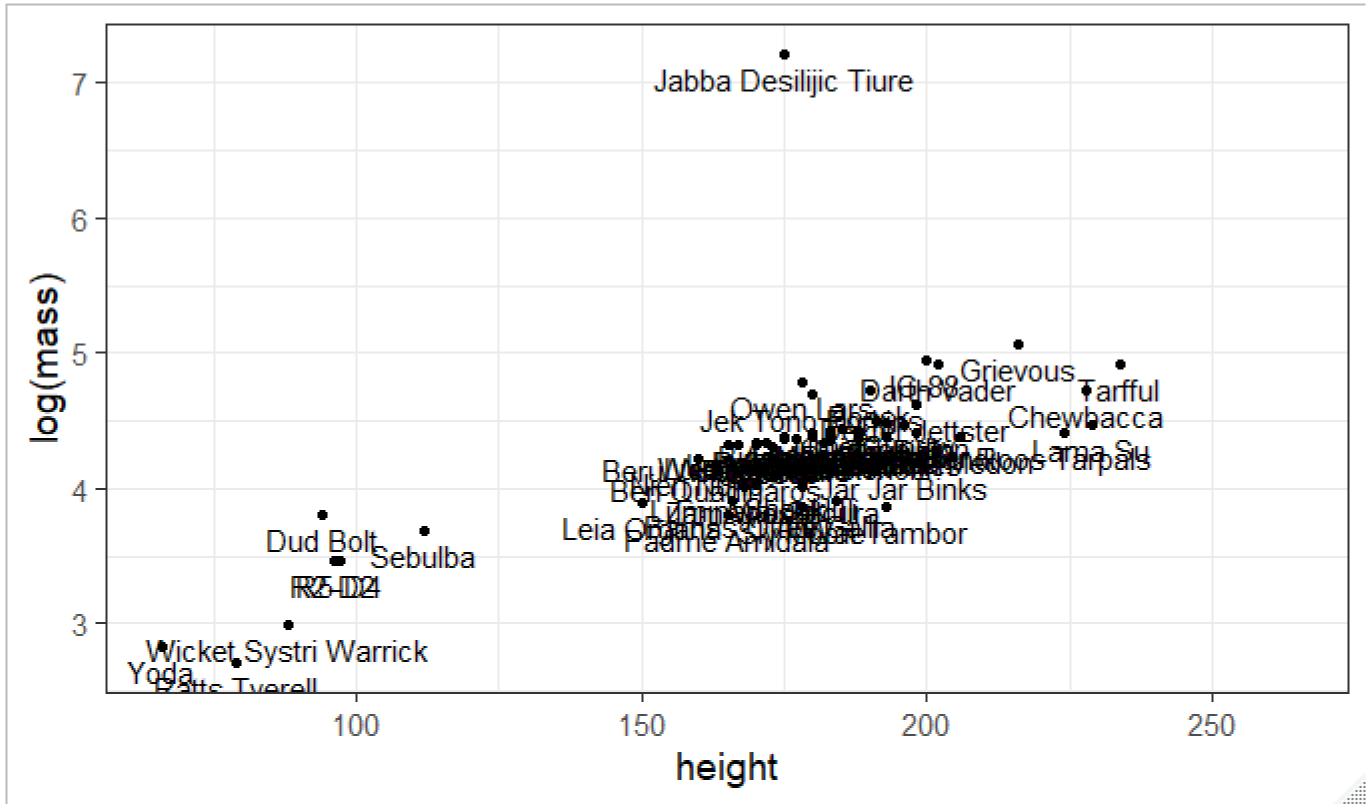
Selective Overlays and Highlighting

Babraham Bioinformatics

Selective highlighting

```
starwars %>%
  ggplot(aes(x=height,y=log(mass), label=name))+
  geom_point() +
  geom_text(vjust=1.5)
```

```
# A tibble: 87 x 4
   name                height  mass homeworld
   <chr>                <int> <dbl> <chr>
 1 Luke Skywalker         172    77 Tatooine
 2 C-3PO                  167    75 Tatooine
 3 R2-D2                   96    32 Naboo
 4 Darth Vader            202   136 Tatooine
```

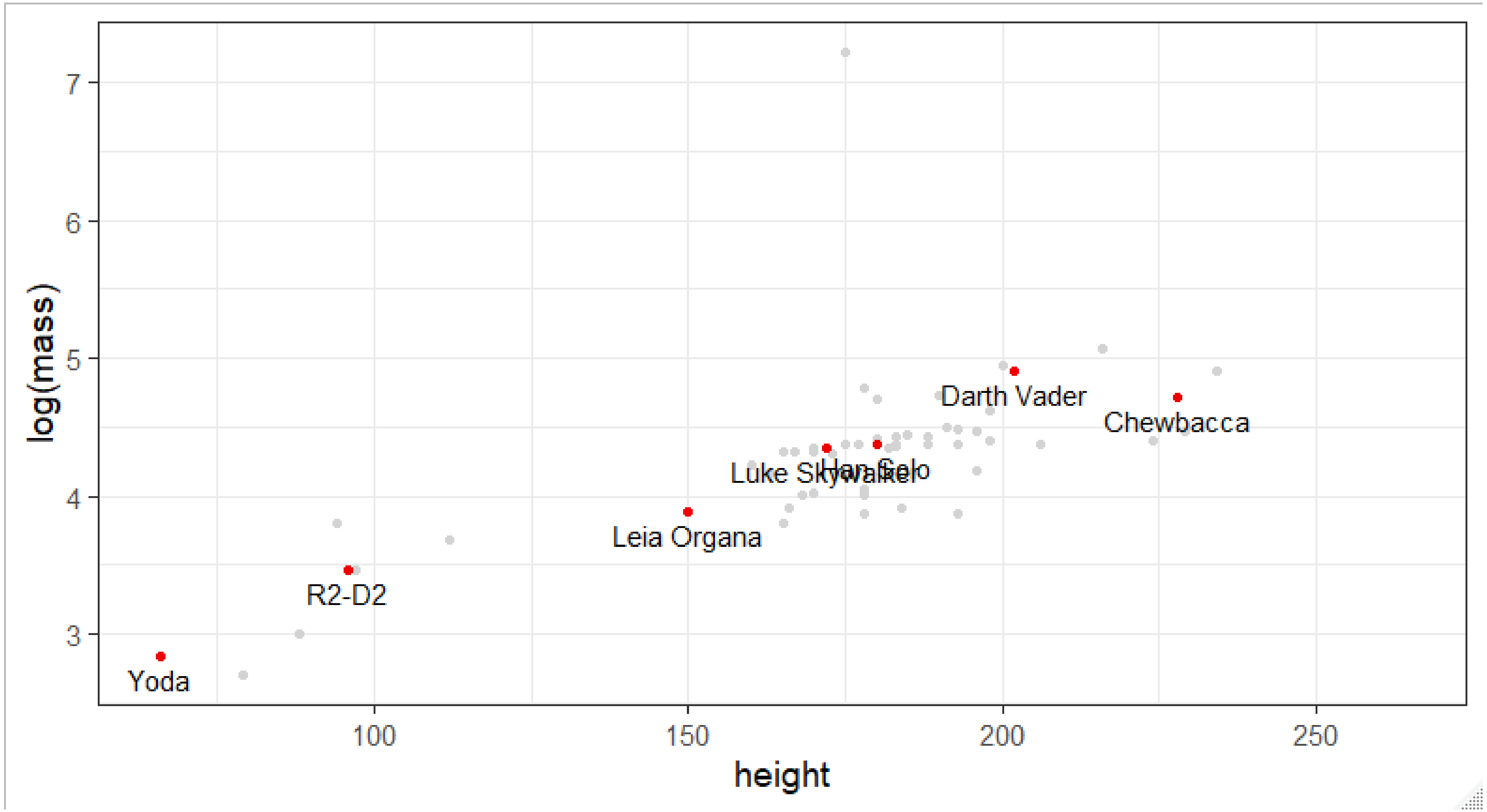


Selective highlighting

```
> famous
[1] "Yoda" "Darth Vader" "Chewbacca" "Han Solo" "R2-D2" "Luke Skywalker" "Leia Organa"
```

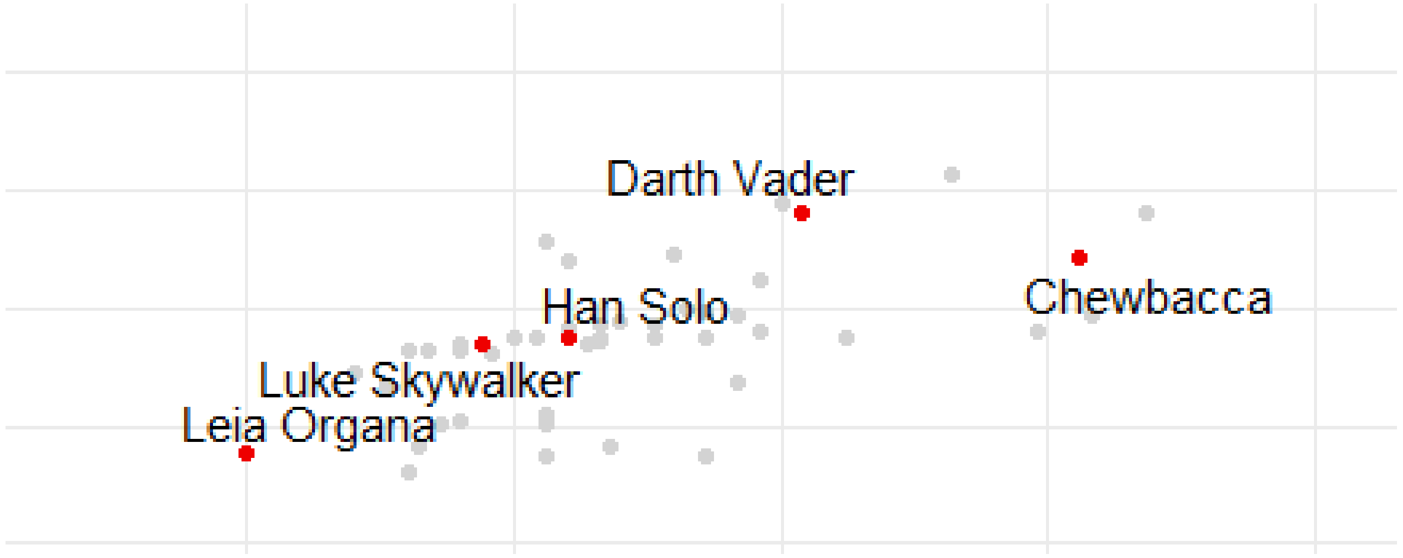
```
starwars %>%
   filter(name %in% famous) -> starwars.famous

starwars %>%
  ggplot(aes(x=height,y=log(mass),label=name))+
  geom_point(col="lightgrey") +
  geom_text(data=starwars.famous)+
  geom_point(data=starwars.famous, color="red2")
```

Selective highlighting



Selective highlighting - ggrepel

```
library(ggrepel)

starwars %>%
  ggplot(aes(x=height,y=log(mass),label=name))+
  geom_point(col="lightgrey") +
  geom_text_repel(data=starwars.famous)+
  geom_point(data=starwars.famous, color="red2")
```



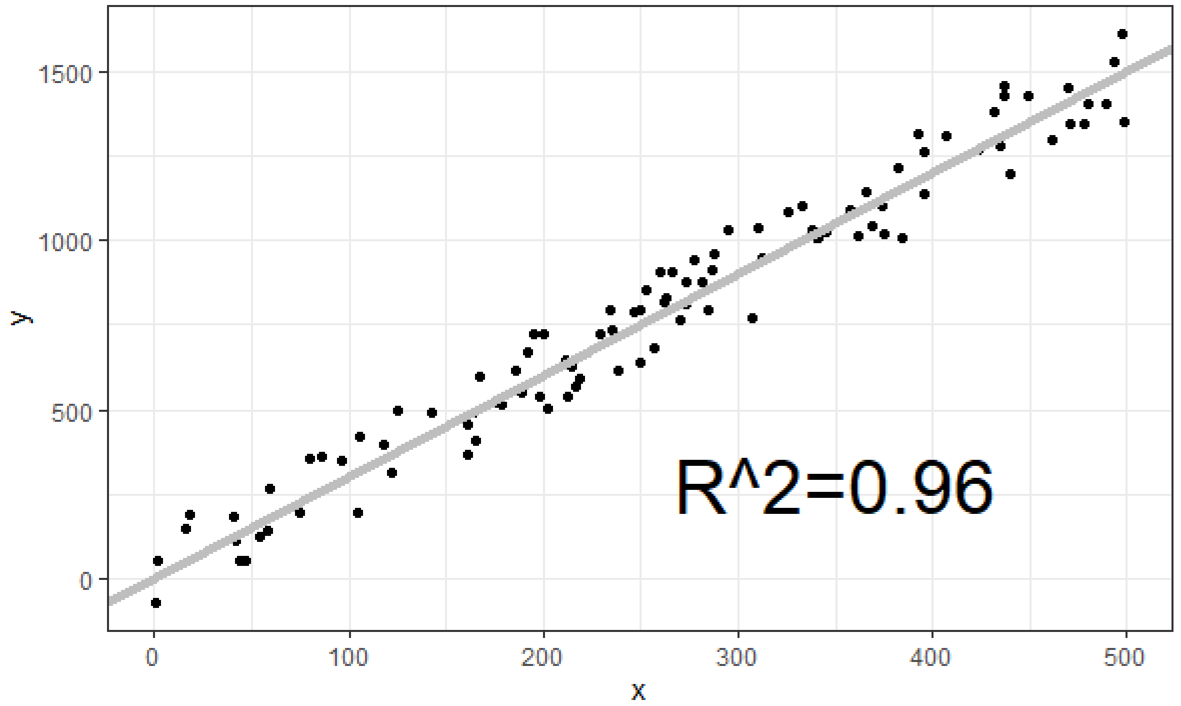
Annotation

- The `annotate` function allows you to add geometries with data not coming from the original tibble

- An easy way to put small additions (usually text) on your plot
  - Aesthetic mappings are set by arguments to `annotate`
  - Values can be multi-element vectors (unlike fixed aesthetics)
  - Use the `geom` argument to say which geometry to use

Annotation

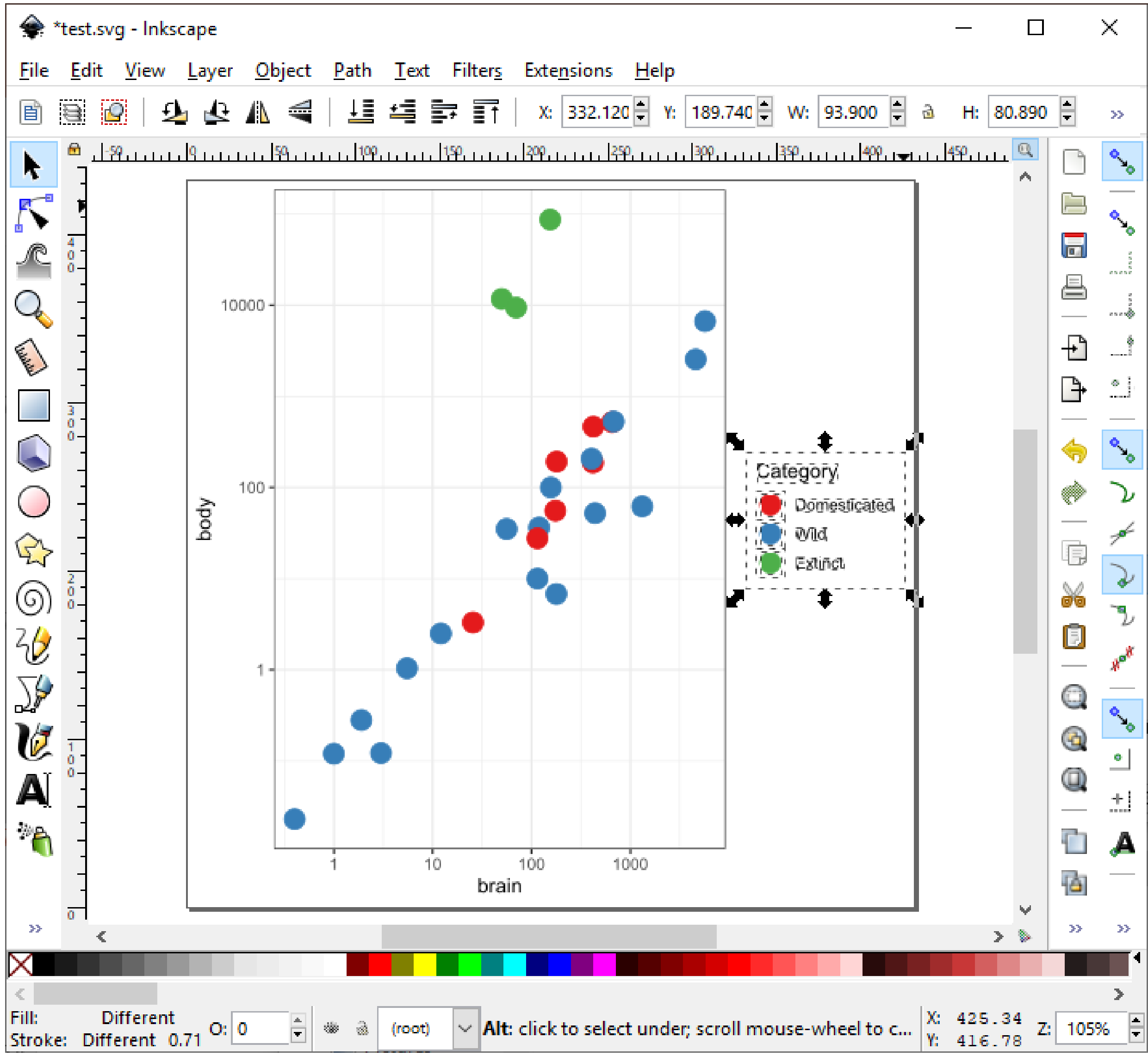
```
data %>%
  ggplot(aes(x=x,y=y)) +
  geom_point() +
  geom_abline(slope = 3, intercept=0, size=2, colour="grey") +
  annotate(geom="text",x=350,y=280,label="R^2=0.96", size=10)
```



Saving plots

- Operates on the last drawn plot by default

```
ggsave(
   filename = "test.svg",
   device = "svg",
   width = 6,
   height = 6
)
```

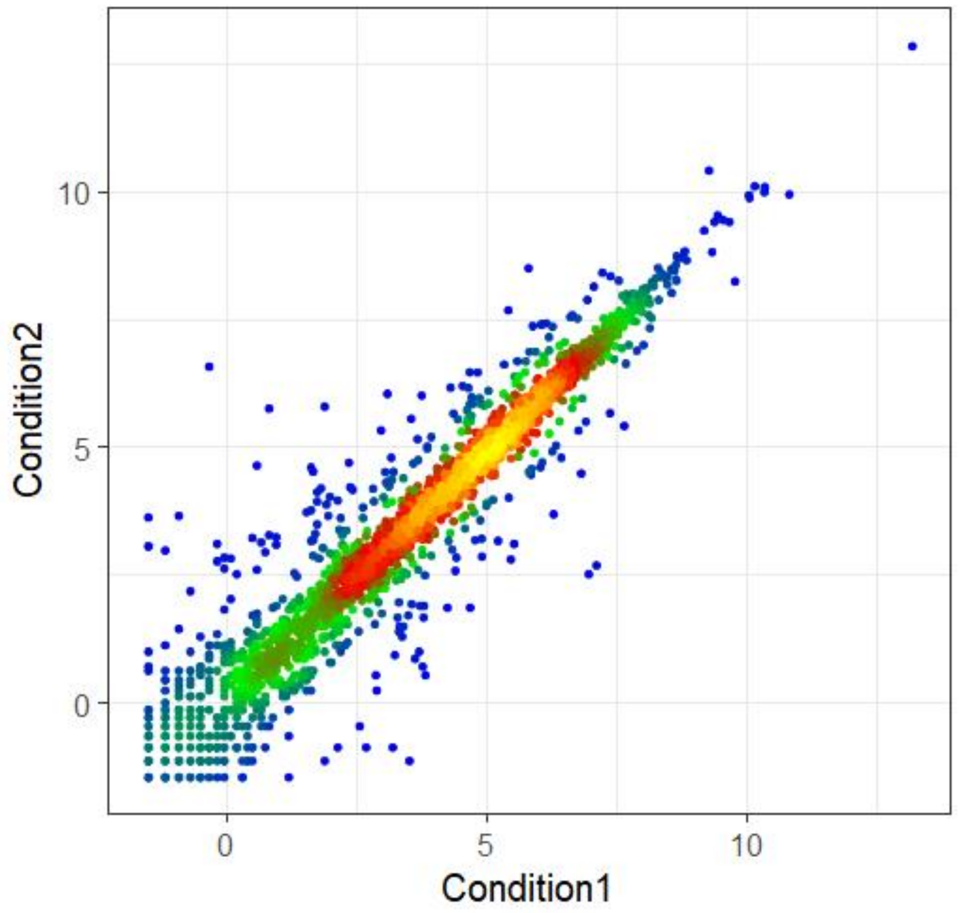


X: 332.120 Y: 189.740 W: 93.900 H: 80.890

Fill: Different
Stroke: Different 0.71
O: 0
(root)
Alt: click to select under; scroll mouse-wheel to c...
X: 425.34
Y: 416.78
Z: 105%

Saving complex plots

```
library(ggrastr)


up_down %>%
  ggplot(aes(x=Condition1,y=Condition2)) +
  rasterise(
    geom_point(color=up_down$colour, size=0.7),
    dpi=150
  ) -> mixplot


ggsave("mixplot.svg", plot=mixplot, device="svg", width=6, height=6)
```



Exercise 5

Babraham Bioinformatics