Generation of high quality graphics for publication with R and ggplot2

Lars Roed Ingerslev

December 3, 2015

Before we start

Required Skills

- Some understanding of R
- How to import data into R
- How to install libraries

Getting the examples

Download from git:

```
git clone https://github.com/LarsRI/gg2.git
```

Useful links

cookbook-r.com/Graphs/index.html
docs.ggplot2.org/current
stackoverflow.com/tags/ggplot2/info

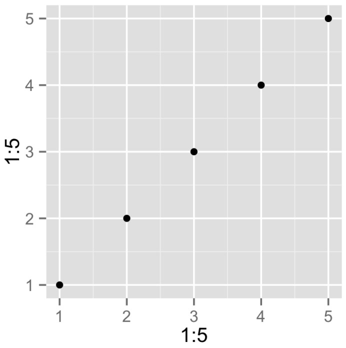
What is ggplot2?

- gg is short for grammar of graphics
- Developed by Hadley Wickham
- Figures are built in layers

qplot

code

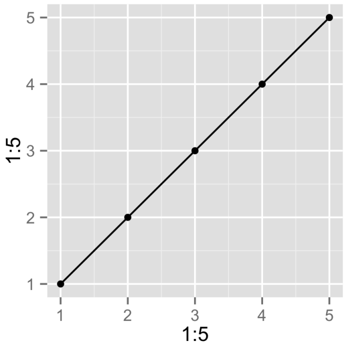
```
p <- qplot(1:5, 1:5)
p
```



qplot

code

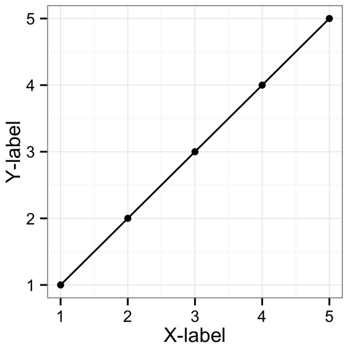
```
p + geom_line()
```



qplot

code

```
p + geom_line() +
    xlab("X-label") +
    ylab("Y-label") +
    theme_bw()
```



Wide format

	Sample1	Sample2	Sample3
Gene1	1	2	3
Gene2	4	5	6
Gene3	7	8	9

Long format

Gene	variable	value
Gene1	Sample1	1
Gene2	Sample1	2
Gene3	Sample1	3
Gene1	Sample2	4
...	...	...
Gene2	Sample3	8
Gene3	Sample3	9

reshape2

code

```
library(reshape2)
ex <- data.frame(matrix(1:9, nrow = 3))
colnames(ex) <- c("Sample1", "Sample2",
                  "Sample3")
ex$Gene <- c("Gene1", "Gene2", "Gene3")
ex
melt(ex)
```

The diamonds dataset

carat	cut	color	clarity	depth	table	price
0.23	Ideal	E	SI2	61.5	55	326
0.21	Premium	E	SI1	59.8	61	326
0.23	Good	E	VS1	56.9	65	327
0.29	Premium	I	VS2	62.4	58	334
0.31	Good	J	SI2	63.3	58	335
0.24	Very Good	J	VVS2	62.8	57	336

The ggplot function

code

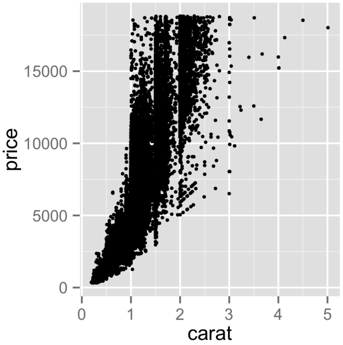
```
p <- ggplot(data = diamonds,
            mapping = aes(x = carat,
                          y = price)
            )
```

ggplot() contains things common to all layers

The ggplot function

code

```
p + geom_point()
```



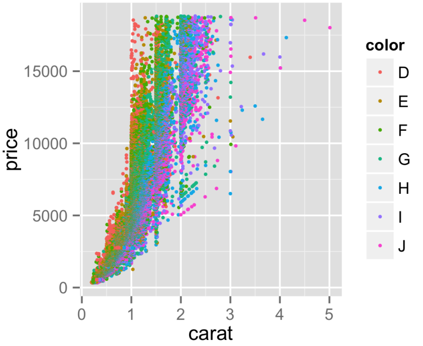
`aes()` defines x, y, colours, shapes etc.

x	X-axis
y	Y-axis
colour	Colour of lines and points
fill	Fill of object
group	Group data by
linetype	Select linetype by
shape	Select point shape by
size	Set size of lines and points by
alpha	Set alpha by

Setting colour

code

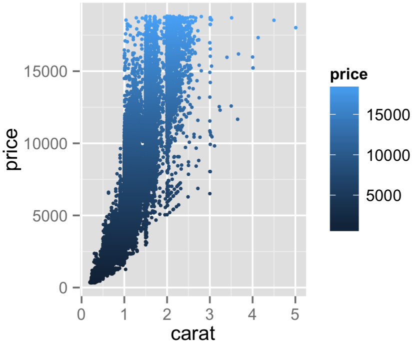
```
p %+%
  aes(colour = color) +
  geom_point()
```



Setting colour

code

```
p %+%
  aes(colour = price) +
  geom_point()
```

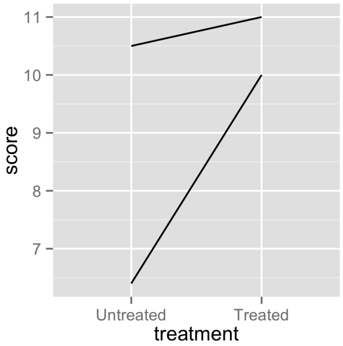


condition	treatment	score
Healthy	Untreated	10.50
Healthy	Treated	11.00
Diseased	Untreated	6.40
Diseased	Treated	10.00

geom_line()

code

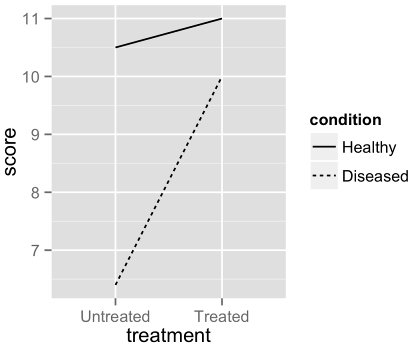
```
p <- ggplot(dat_gl,
       aes(
         x = treatment,
         y = score,
         group =
           condition
       ))
p + geom_line()
```



geom_line()

code

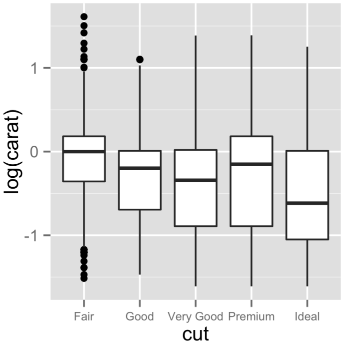
```
p %+% aes(
  linetype = condition) +
    geom_line()
```



geom_boxplot()

code

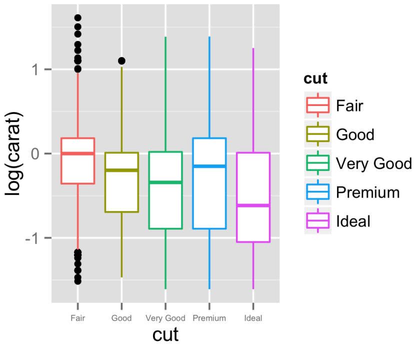
```
p <- ggplot(diamonds,
        aes(x = cut,
          y = log(carat)
          )
        )
```



geom_boxplot()

code

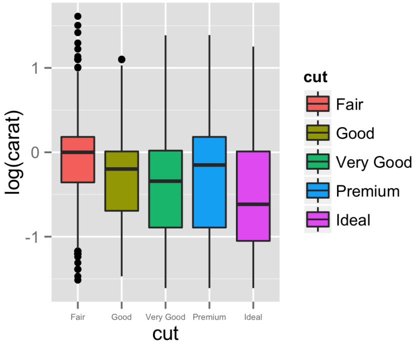
```
p + geom_boxplot(
        aes(colour = cut)
    )
p + geom_boxplot()
```



geom_boxplot()

code

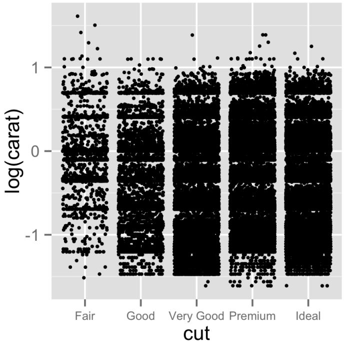
```
p + geom_boxplot(
      aes(fill = cut)
    )
```



geom_jitter()

code

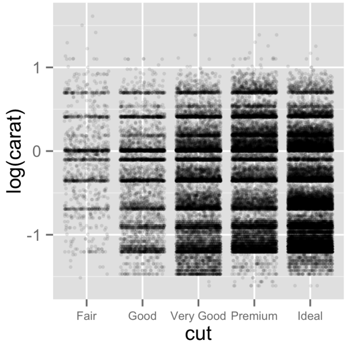
```
p + geom_jitter()
```



geom_jitter()

code

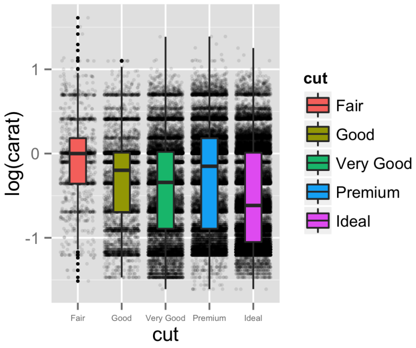
```
p + geom_jitter(
      alpha = 0.1
    )
```



combined

code

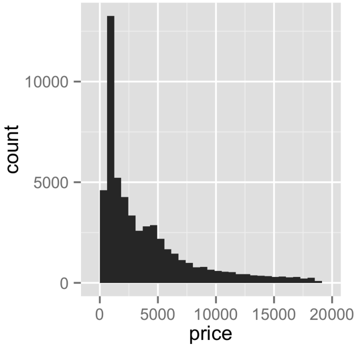
```
p + geom_jitter(
      alpha = 0.1
    ) +
    geom_boxplot(
        aes(fill = cut)
        width = 0.5
    )
```



geom_histogram()

code

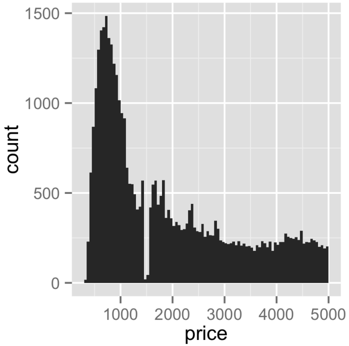
```
p <- ggplot(diamonds,
        aes(x = price))
p + geom_histogram()
```



geom_histogram()

code

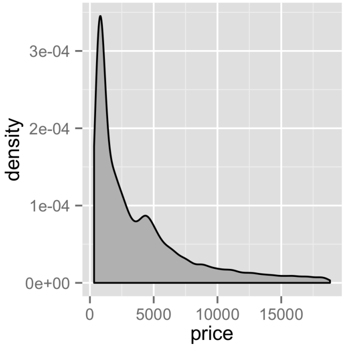
```
p + geom_histogram(
      binwidth = 50
    ) +
    xlim(c(300, 5000))
```



geom_histogram()

code

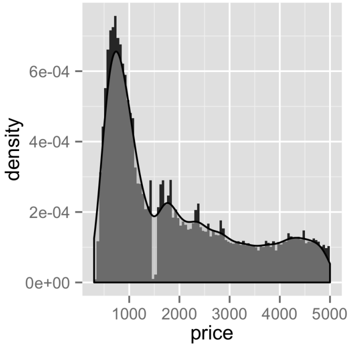
```
p + geom_density(
      fill = "grey"
    )
```



geom_histogram()

code

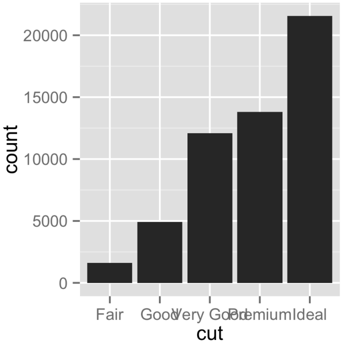
```
p + geom_histogram(
      aes(y =
        ..density..),
    binwidth = 50) +
    geom_density(
      fill = "grey",
      alpha = 0.5
    ) + xlim(c(300, 5000))
```



geom_bar()

code

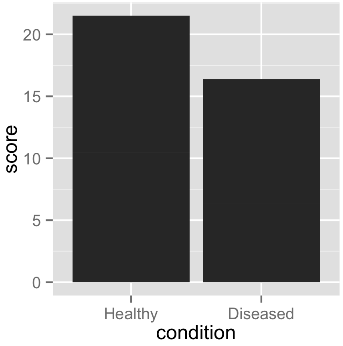
```
ggplot(diamonds,
        aes(x = cut)) +
       geom_bar()
```



geom_bar() + geom_errorbar()

code

```
p <- ggplot(dat_gl,
  aes(x = condition,
      y = score,
      group = treatment
  )
p + geom_bar(
      stat = "identity"
    )
```



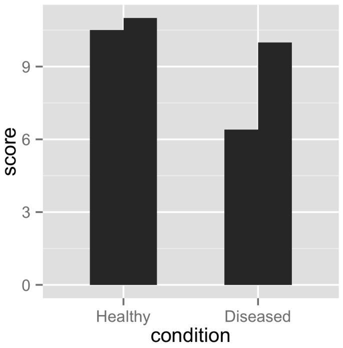
`position` refines the position of objects, is a function `position_*`

identity	Do nothing
stack	Stack objects
fill	Stack objects, and force equal height
jitter	Randomly move object a little
dodge	Move overlapping objects side by side
jitterdodge	Move overlapping objects side by side and then randomly move object a little

geom_bar() + geom_errorbar()

code

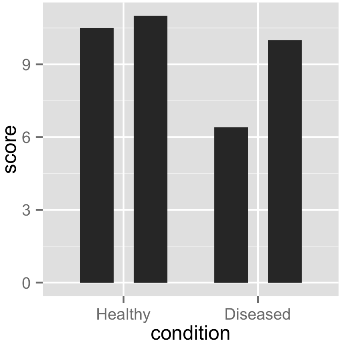
```
p + geom_bar(
    stat = "identity",
    position = "dodge",
    width = 0.5
    )
```



geom_bar() + geom_errorbar()

code

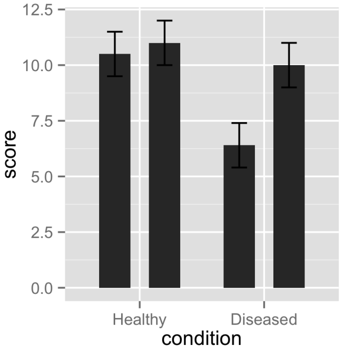
```
p <- p + geom_bar(
  stat = "identity",
  position =
    position_dodge(0.8)
  width = 0.5)
p
```



geom_bar() + geom_errorbar()

code

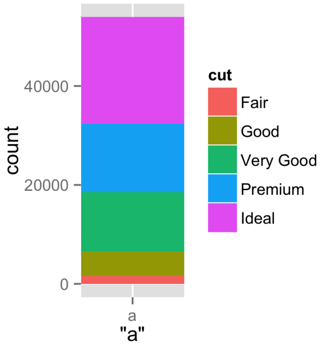
```
p + geom_errorbar(
  aes(ymin = score - 1,
      ymax = score + 1)
  position =
    position_dodge(0.8)
  width = 0.25)
```



Pie chart

code

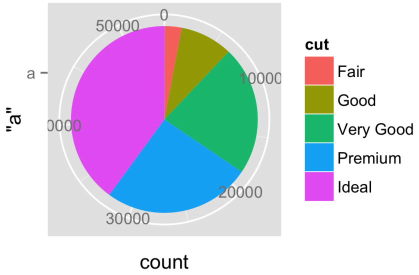
```
p <- ggplot(diamonds,
       aes(x = 'a',
           fill = cut)
    ) +
    geom_bar(
      width = 1
    )
p
```



Pie chart

code

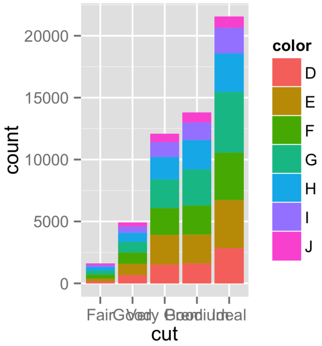
```
p + coord_polar(
      theta = 'y'
    )
```



geom_bar()

code

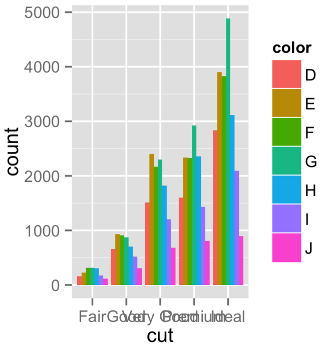
```
p <- ggplot(diamonds,
        aes(x = cut,
        fill = color)
      )
p + geom_bar()
```



geom_bar()

code

```
p + geom_bar(
      position =
        "dodge"
    )
```



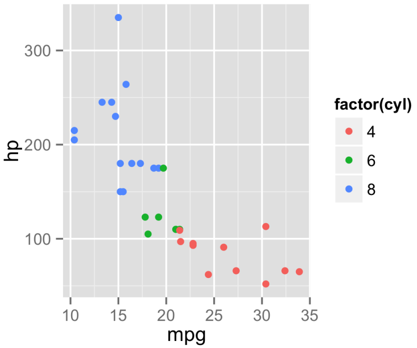
The mtcars dataset

	mpg	cyl	disp	hp
Mazda RX4	21.00	6.00	160.00	110.00
Mazda RX4 Wag	21.00	6.00	160.00	110.00
Datsun 710	22.80	4.00	108.00	93.00
Hornet 4 Drive	21.40	6.00	258.00	110.00
Hornet Sportabout	18.70	8.00	360.00	175.00
Valiant	18.10	6.00	225.00	105.00

faceting

code

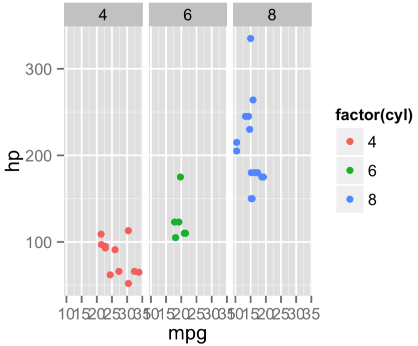
```
p <- ggplot(mtcars,
  aes(mpg,
      hp,
      colour =
          factor(cyl)))
    geom_point()
p
```



faceting

code

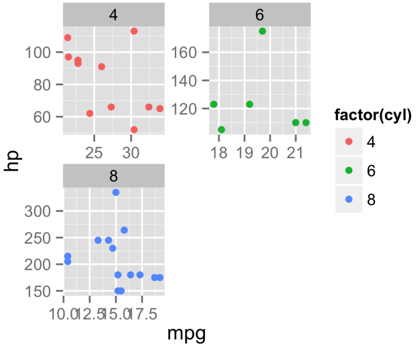
```
p + facet_wrap(~cyl)
```



faceting

code

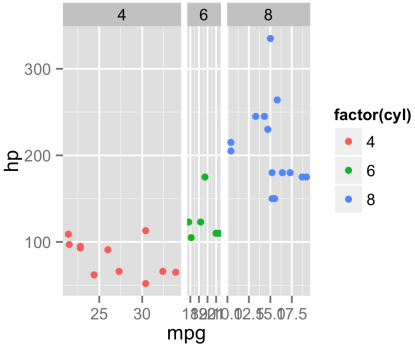
```
p + facet_wrap(~cyl,
        scales = "free",
        nrow = 2)
```



faceting

code

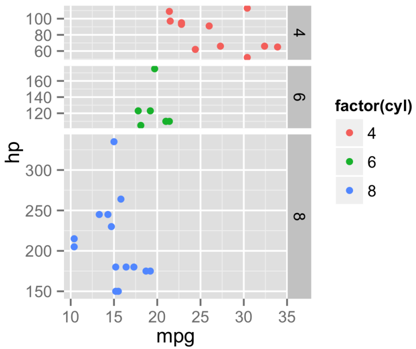
```
p + facet_grid(~cyl,
        scales = "free",
        space = "free")
```



faceting

code

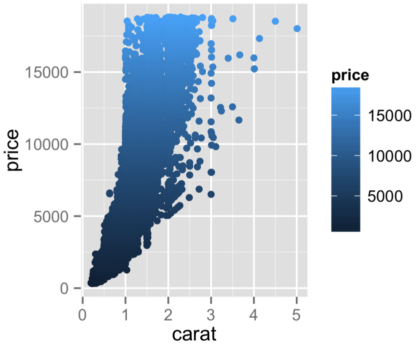
```
p <- p + facet_grid(
      cyl~.,
      scales = "free",
      space = "free")
p
```



Continous colour scales

code

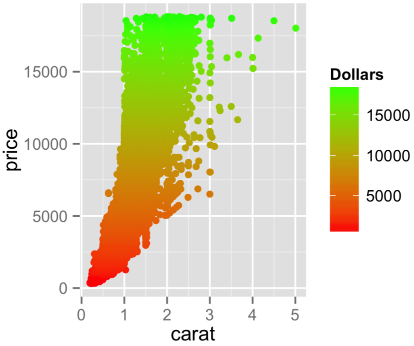
```
p <- ggplot(diamonds,
       aes(carat,
         price,
         colour =
           price)
     ) +
    geom_point()
p
```



Continous colour scales

code

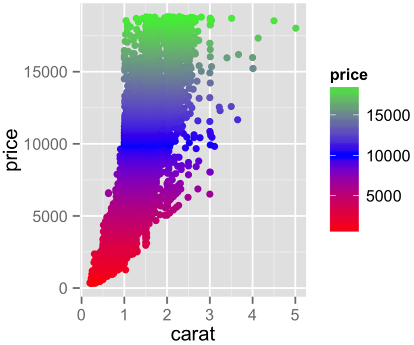
```
p +
 scale_colour_gradient(
   name = "Dollars",
    low = "red",
    high = "green")
```



Continous colour scales

code

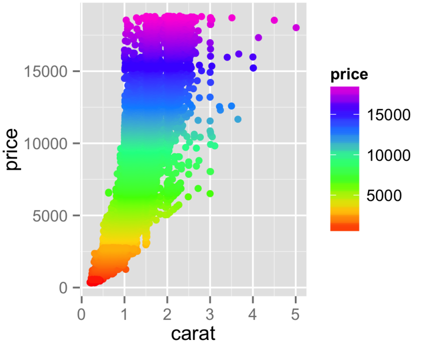
```
p +
scale_colour_gradient2(
    low = "red",
    mid = "blue",
    high = "green",
    midpoint = 10000)
```



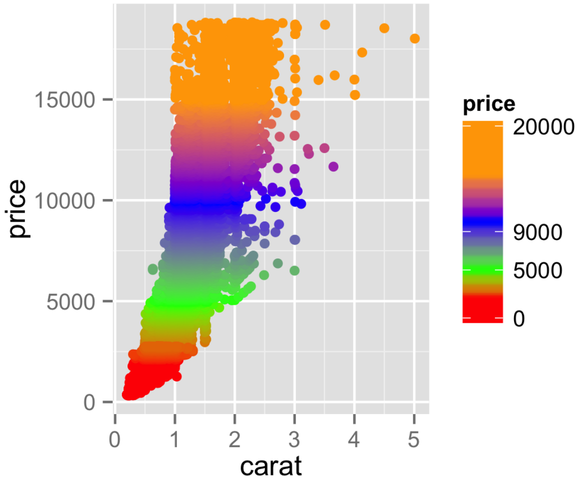
Continous colour scales

code

```
p +
scale_colour_gradientn(
  colours = rainbow(7)
  )
```



Continous colour scales



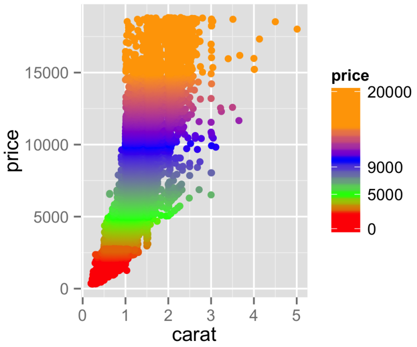
Continous colour scales

```
colScale <-
 scale_colour_gradientn(
  rescaler = function(x, ...)x,
  oob = identity,
  colours=c("red", "red", "green",
            "blue", "orange", "orange"),
  values=c(0, 2000, 5000, 10000,
            15000, 20000),
  breaks = c(0, 5000, 9000, 20000),
  limits = c(0, 20000))
```

Continous colour scales

code

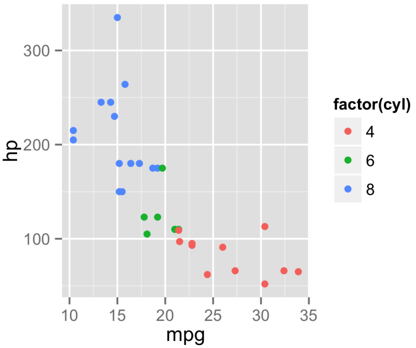
```
p +
  colScale
```



Manual colour scales

code

```
p <- ggplot(mtcars,
  aes(mpg,
      hp,
      colour =
          factor(cyl)))
    geom_point()
p
```



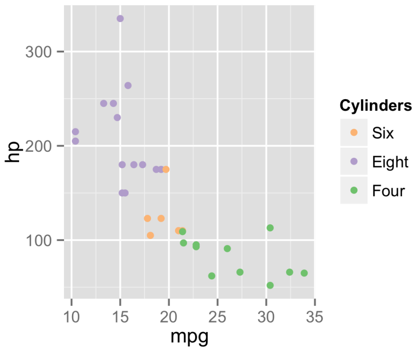
Manual colour scales

```
colScale <-
    scale_colour_manual(
        values = c('4' = "#7fc97f",
        '8' = "#beaed4",
        '6' = "#fdc086"),
        labels = c('4' = "Four", '6' = "Six",
                   '8' = "Eight"),
        breaks = c('6', '8', '4'),
        name = "Cylinders"
        )
```

Manual colour scales

code

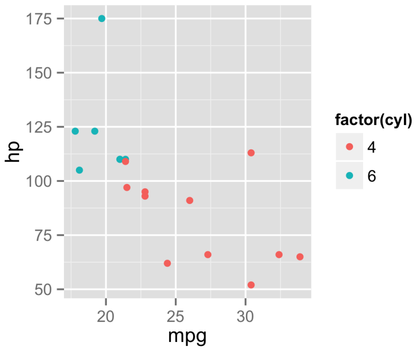
```
p + colScale
```



Manual colour scales

code

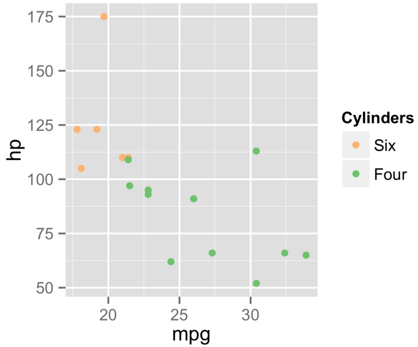
```
p %+% subset(
         mtcars,
         cyl != 8
      )
```



Manual colour scales

code

```
p %+% subset(
        mtcars,
        cyl != 8
     ) +
  colScale
```



Manual colour scales

```
colScale <-
    scale_colour_manual(
        values = c('4' = "#7fc97f",
        '8' = "#beaed4",
        '6' = "#fdc086"),
        labels = c('4' = "Four", '6' = "Six",
                   '8' = "Eight"),
        breaks = c('6', '8', '4'),
        name = "Cylinders",
        drop = FALSE)
```

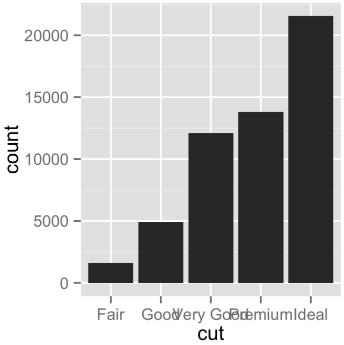
Manual fill scales

```
scale_fill_gradient()
scale_fill_gradient2()
scale_fill_gradientn()
scale_fill_manual()
```

Axis scales

code

```
p <- ggplot(diamonds,
            aes(x = cut)) +
           geom_bar()
p
```



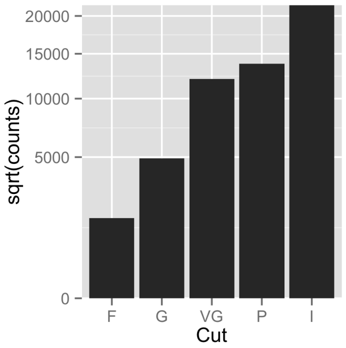
scale_x_discrete()

```
xScale <- scale_x_discrete(name = "Cut",
      labels = c("Fair" = "F",
                 "Good" = "G",
                 "Very Good" = "VG",
                 "Premium" = "P",
                 "Ideal" = "I"))
yScale <- scale_y_sqrt("sqrt(counts)",
expand = c(0,0))
```

Axis scales

code

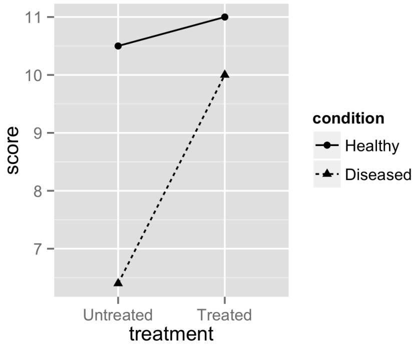
```
p + xScale + yScale
```



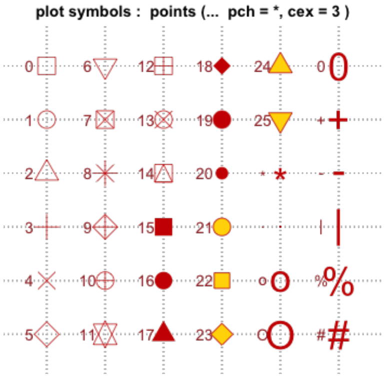
Points and lines

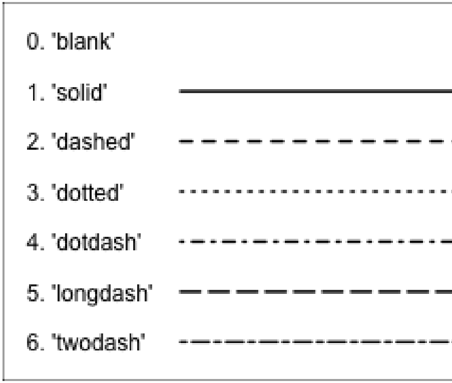
code

```
p <- ggplot(dat_gl,
  aes(x = treatment,
    y = score,
    shape = condition,
    linetype = condition,
    group = condition)) +
  geom_point() +
  geom_line()
p
```



Points and lines





cookbook-r.com/Graphs/Shapes_and_line_types/

scale_linetype_discrete()

```
lineScale <- scale_linetype_manual(
    values = c("Healthy" = 1, "Diseased" = 2)
    labels = c('Healthy' = "A-OK",
               'Diseased' = "Not so good"),
    breaks = c('Healthy', 'Diseased'),
    name = "Types of object"
)
```

scale_linetype_discrete()

```
lineScale2 <- scale_linetype_manual(
    values = c("Healthy" = 1, "Diseased" = 2)
)
```

scale_shape_discrete()

```
shapeScale <- scale_shape_manual(
    values = c("Healthy" = 1, "Diseased" = 2)
)
```

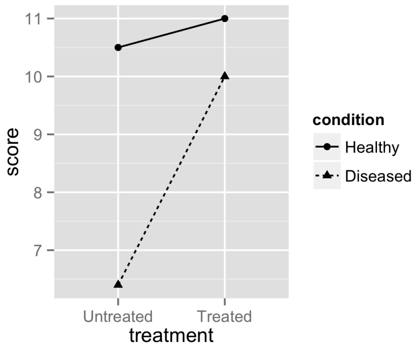
scale_alpha_*

```
scale_alpha_continuous()
scale_alpha_manual()
```

Removing guides

code

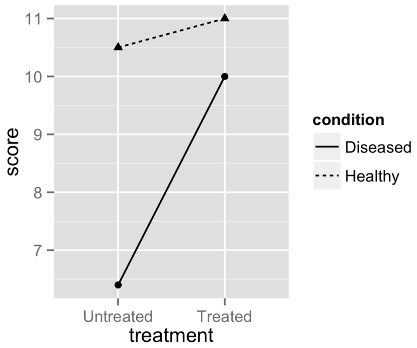
p



Removing guides

code

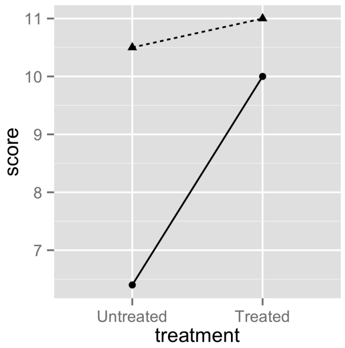
```
p + guides(
  shape = FALSE
)
```



Removing guides

code

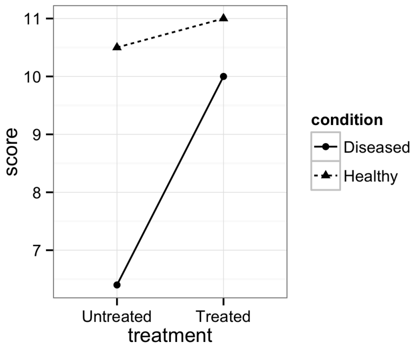
```
p + guides(
  shape = FALSE,
  linetype = FALSE
)
```



Themes

code

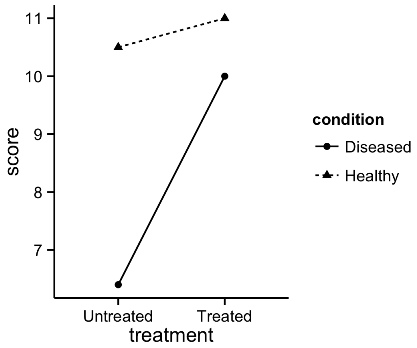
```
p + theme_bw()
```



Themes

code

```
p + theme_classic()
```



Different elements

element_rect()	Rectangles: backgrounds, legend box, etc.
element_text()	Texts: axes, legend names, titles, etc.
element_line()	Lines: one the plot, ticks, edges, etc.
element_blank()	Removes this element.

theme()

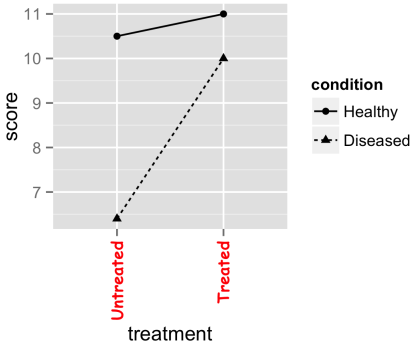
theme() arguments are hierarchical, see documentation for all arguments.

text	Affects all text on the plot
axis.text	Affects only the text on the axes
axis.text.x	Affects only the text on the x-axis

Themes

code

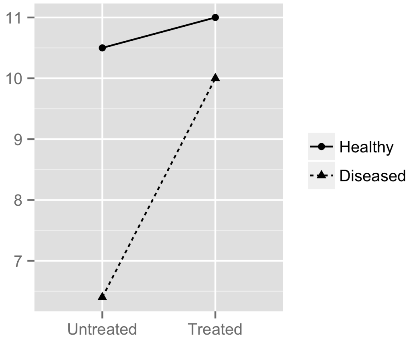
```
p + theme(axis.text.x =
 element_text(
  family =
    "Comic Sans MS",
  colour = "red",
  angle = 90,
  face = "bold.italic",
  vjust = 0.5, hjust = 1
))
```



Themes

code

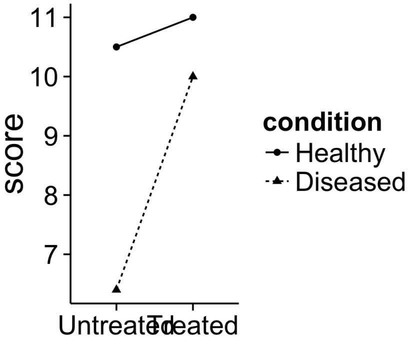
```
p + theme(
    legend.title =
        element_blank()
    axis.title =
        element_blank()
)
```



Themes

code

```
p + theme_classic(
    base_size = 20,
    base_family =
      "Arial") +
    theme(
      axis.title.x =
        element_blank()
      )
```



Random Heatmap

```
set.seed(1)
mat <- matrix(rnorm(600), ncol = 6)
colnames(mat) <-
    paste(rep(c("p1", "p2", "p3"), 2),
          rep(c("baseline", "treated"),
              each = 3))
rownames(mat) <- paste0("Gene",
                    seq_len(nrow(mat)))
```

Random Heatmap

```
mat2 <- melt(mat)
mat2$condition <- sub("p\\d ",
                      "", mat2$Var2)
mat2$Var2 <- sub("(p\\d).+",
                      "\\1", mat2$Var2)
colnames(mat2) <- c("Gene", "Participant",
                       "Score", "Condition")
mat2$Significant <- abs(mat2$Score) > 2.5
```

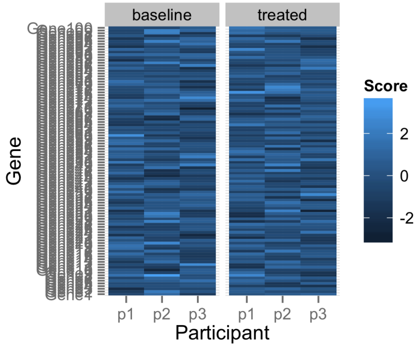
Random Heatmap

Gene	Participant	Score	Condition	Significant
Gene1	p1	-0.63	baseline	FALSE
Gene2	p1	0.18	baseline	FALSE
Gene3	p1	-0.84	baseline	FALSE
Gene4	p1	1.60	baseline	FALSE
Gene5	p1	0.33	baseline	FALSE
Gene6	p1	-0.82	baseline	FALSE

Random Heatmap

code

```
p <- ggplot(mat2, aes(
        x = Participant,
        y = Gene,
        fill = Score)) +
    geom_tile() +
    facet_wrap(
      ~Condition)
p
```



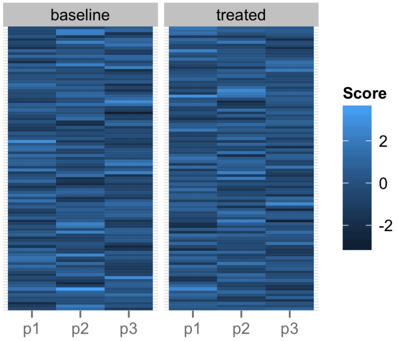
Random Heatmap

```
myTheme <- theme(
  axis.text.y = element_blank(),
  axis.title = element_blank(),
  axis.ticks.y = element_blank()
  )
```

Random Heatmap

code

```
p + myTheme
```



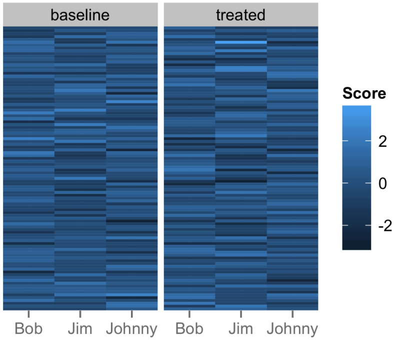
Random Heatmap

```
xScale <- scale_x_discrete(
   labels = c("p1" = "Bob",
                     "p2" = "Jim",
                     "p3" = "Johnny"),
  expand = c(0,0)
  )
```

Random Heatmap

code

```
p + myTheme +
    xScale
```



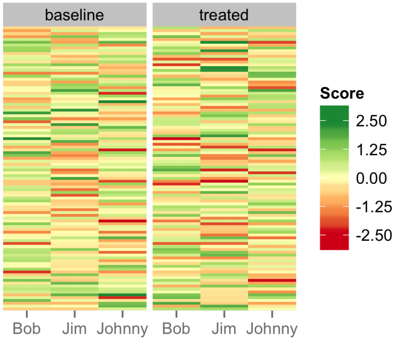
Random Heatmap

```
fillScale <-
  scale_fill_gradientn(colours=
    c("#d7191c", "#d7191c", "#fdae61",
    "#ffffbf", "#a6d96a", "#1a9641",
    "#1a9641"
    ), rescaler = function(x, ...)x,
    oob = identity,
    values=c(-5, -2.5, -1.25, 0, 1.25, 2.5, 5
    breaks = c(-2.5, -1.25, 0, 1.25, 2.5),
    limits = c(-3, 3))
```

Random Heatmap

code

```
p + myTheme +
    xScale +
    fillScale
```



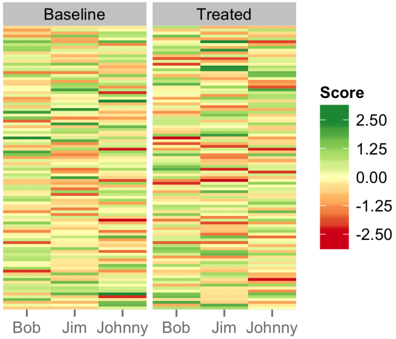
Random Heatmap

```
labelConversion <- list(
    "baseline" = "Baseline",
    "treated" = "Treated"
    )
c_labeller <- function(variable,value){
    return(labelConversion[value])
}
```

Random Heatmap

code

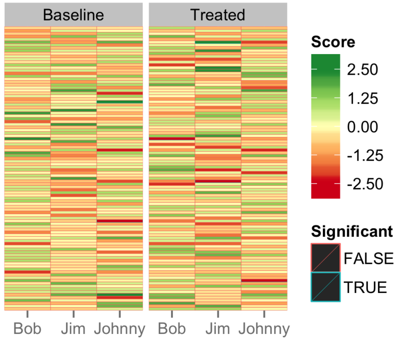
```
p <- p + myTheme +
 xScale +
 fillScale +
 facet_grid(~Condition,
 labeller=c_labeller
 )
p
```



Random Heatmap

code

```
p %+% aes(
  colour = Significant
  )
```



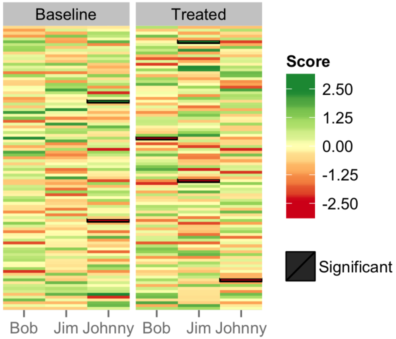
Random Heatmap

```
colScale <- scale_colour_manual(
  name = element_blank(),
  values = c("TRUE" = "black", "FALSE" = NA),
  labels = c("TRUE" = "Significant"),
  limits = "TRUE")
```

Random Heatmap

code

```
p %+%
  aes(
    colour = Significan
    ) +
  colScale +
  geom_tile(size = 0.5)
```



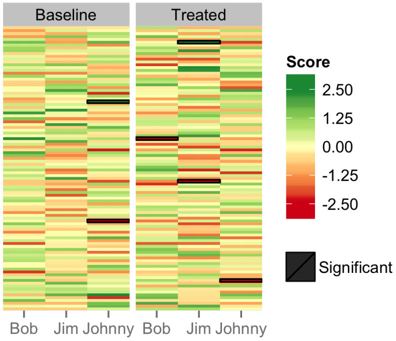
Random Heatmap

```
alphaScale <- scale_alpha_manual(
  name = element_blank(),
  values = c("TRUE" = 1, "FALSE" = 0),
  guide = FALSE)
```

Random Heatmap

code

```
p + geom_tile(
 aes(
  colour = Significant,
  alpha = Significant),
  size = 0.5) +
  colScale +
  alphaScale
```



What's next?

grid + gridExtra	Packages for arranging multiple plots
annotate	Function for arbitrary annotations
other geoms	I only covered some of them
stat_*	I only briefly touched on them

Download from git:

```
git clone https://github.com/LarsRI/gg2.git
```