# ggplot Reference Card

by Dag Ø. Hjermann, NIVA, Norway ([dhj@niva.no](mailto:dhj@niva.no))

df = name of data frame  
xvar, yvar, zvar = names of variables

## Scatter plots

ggplot(df, aes(x=xvar, y=yvar)) + geom\_point()

Change to red cross symbols

... + geom\_point(colour = "red", size = 3,   
shape=3)

Change to large overlapping circles

... + geom\_point(colour = "red", fill="blue", shape=21, size = rel(15), alpha = 0.6)

Scatter with colour for continuous/categorical z variable

ggplot(df, aes(x=xvar, y=yvar, color=z)) + geom\_point()

…exactly same result with:

ggplot(df, aes(x=xvar, y=yvar)) + geom\_point(aes(color=zvar))

If zvar is given as numbers, but you want categorical colors:

ggplot(df, aes(x=xvar, y=yvar, color=factor(zvar))) + geom\_point()

Coloured points with black border

ggplot(df, aes(x=xvar, y=yvar, fill=zvar)) + geom\_point(shape=21)

Options for continuous z variables

... + scale\_colour\_gradient(low = "blue")

Options for categorical z variables

... + scale\_colour\_brewer(palette="Dark2")

Scatter plot with symbol size/shape for z variable

... + geom\_point(aes(size=zvar))

... + geom\_point(aes(shape=zvar))

... + scale\_shape(solid = FALSE) # open symbols

Jittering points to reduce overplotting

... + geom\_point(position = "jitter")

or

... + geom\_jitter(aes(color=zvar))

Jittering in only one direction (here, x direction)

... + geom\_point(position =   
position\_jitter(width = .5, height = 0))

## Line plots

Connect points ordered by x value

ggplot(df, aes(x=xvar, y=yvar)) + geom\_line()

Connect points ordered by order in data set

ggplot(df, aes(x=xvar, y=yvar)) + geom\_path()

Several lines defined by variable "zvar"

ggplot(df, aes(x=xvar, y=yvar, group=zvar)) + geom\_line(aes(color=zvar))

Several lines with error bars (when S.E. is known)

limits <- aes(ymax = yvar + se, ymin=yvar – se, color=zvar)

ggplot(df, aes(x=xvar, y=yvar, group=zvar)) +  
geom\_line(aes(color=zvar)) +  
geom\_errorbar(limits, width = 0.2)

Change colour, line thickness, line type and transparency for all lines

... + geom\_line(color = "red", size = 2,   
linetype = "dashed", alpha = 0.8)

Some allowed line types:

"solid","dashed","dotted","dotdash","longdash"

Several lines varying by colour or thickness (continuous)

... + geom\_line(aes(colour = zvar))

... + geom\_line(aes(size = zvar))

Several lines varying by line type (categorical)

... + geom\_line(aes(linetype = zvar))

Several lines varying according to two categorical variables (zvar1 and zvar2), shown by line type and point type

ggplot(df5, aes(x=xvar, y=yvar,  
group=interaction(zvar1,zvar2))) + geom\_line(aes(linetype = zvar1)) + geom\_point(aes(shape=zvar2))

Line thickness varies along the line (e.g. time series)

ggplot(df, aes(x=xvar, y=yvar)) + geom\_line(aes(size = zvar))

## Ribbon plots

## Adding fitted lines

Default: loess (or gam for large datasets) with CI:

... + stat\_smooth()

No confidence interval:

... + stat\_smooth(se = FALSE)

Linear regression

... + stat\_smooth(method = "lm")

Fit data

... + stat\_smooth(method = "lm")

Change line (for more details see "line plots")

... + stat\_smooth(colour="darkblue", size=2))

Change confid. interval color and transparancy

...stat\_smooth(fill = "blue", alpha = 1)

## Bar plots (for histograms, see below)

Simple bar plot (1 categorical variable)

ggplot(df, aes(x=xvar, y=yvar)) +   
geom\_bar(stat="identity")

If one doesn't set stat="identity" the result is a histogram:

Simple bar plot + precalculated error bars

ggplot(df, aes(x=xvar, y=yvar)) +   
geom\_bar(stat="identity") + geom\_errorbar(aes(ymin=yvar-se, ymax=yvar+se), width = 0.4)

Stacked bar chart

ggplot(df, aes(x=xvar, y=yvar, fill=category)) +   
geom\_bar(stat="identity")

Side-by-side bar chart (within groups set by *xvar*)

ggplot(df, aes(x=xvar, y=yvar, fill=category)) +   
geom\_bar(stat="identity", position="dodge")

Side-by-side bar chart + error bars

dodge <- position\_dodge(width=0.9)

ggplot(df, aes(x=xvar, y=yvar, fill=category)) +   
geom\_bar(stat="identity", position=dodge) +  
geom\_errorbar(aes(ymin=yvar-se, ymax=yvar+se), position="dodge", width = 0.4)

Set colors manually (see "Colors" below):

... + scale\_colour\_manual(values=c("red","blue"))

## Histograms

ggplot(df, aes(x=xvar)) + geom\_histogram()

Bars divided by colours for each category

ggplot(df, aes(x=xvar, fill=category)) + geom\_histogram()

…or:

... + geom\_histogram(aes(fill=category))

Bars side-by-side for each category

ggplot(df, aes(x=xvar, fill=category)) + geom\_histogram(position="dodge")

One plot for each category, stacked

... + facet\_grid(category ~ .)

Set width or number of of bins

... + geom\_histogram(binwidth=5)

... + geom\_histogram(bins=50)

Set y axis to be density instead of count

... + geom\_histogram(aes(y= ..density..))

Set y axis to be percentage of counts

... + geom\_histogram(aes(y= ..count../sum(..count..)))

## Boxplots and violin plots

ggplot(df, aes(y=yvar)) + geom\_boxplot()

ggplot(df, aes(x=xvar, y=yvar)) + geom\_violin()

xvar is already a factor:

ggplot(df, aes(x=xvar, y=yvar)) + geom\_boxplot()

xvar is numeric:

ggplot(df, aes(x=factor(xvar), y=yvar)) + geom\_boxplot()

Split data by another variable in addition to xvar:

ggplot(df, aes(x=xvar, y=yvar)) +   
geom\_boxplot(aes(color=zvar))

## Violin plots, settings

Scale maximum width proportional to sample size:

... + geom\_violin(scale = "count")

Scale maximum width to 1 for all violins:

... + geom\_violin(scale = "width")

## Density plots

As lines

ggplot(df, aes(x=xvar, color=category)) + geom\_density()

As shaded areas

ggplot(df, aes(x=xvar, fill=category)) + geom\_density(alpha=0.2)

## 2D density

ggplot(df, aes(x=xvar, y=yvar)) + stat\_bin2d()

## Logarithmic (and other) axes

... + scale\_x\_log10()

... + scale\_y\_log10()

Add log tick marks (with diminishing spacing)

... + annotation\_logticks(sides = "b")

... + annotation\_logticks(sides = "l")

... + annotation\_logticks(sides = "bl")

Customize tick breaks/labels (here: 100, 101, 102, etc.)

+ scale\_x\_log10(breaks = trans\_breaks("log10", function(x) 10^x),   
labels = trans\_format("log10", math\_format(10^.x)))

Other transformations, example:

... + coord\_trans(y = "sqrt")

Note that these changes are performed after statistics (e.g., a linear regression line will appear curved in the plot)

## Custom axis limits

... + xlim(55, 70)

... + ylim(0, 20)

## Other axis modifications

Flip x and y scale

... + coord\_flip()

Force 1 unit to be equally long on x and y scale

... + coord\_fixed()

Force a specific ratio (y/x) between units

... + coord\_fixed(3)

Reverse scales

... + scale\_x\_reverse()

... + scale\_y\_reverse()

## Faceting

Facet by one or two variables

... + facet\_grid(var1 ~ .) # stacked plots

... + facet\_grid(. ~ var2) # side-by-side

... + facet\_grid(var1 ~ var2) # grid of plots

Facet by one variable, wrapped in 2d on the plot

... + facet\_wrap(~var1)

... + facet\_wrap(~var1, ncol=4) # no. of columns

... + facet\_wrap(~var1, nrow=3) # no. of rows

Default : same x and y scales for all plots. Alternatives:

... + facet\_grid(var1 ~ var2, scales = "free\_x")

... + facet\_grid(var1 ~ var2, scales = "free\_y")

... + facet\_grid(var1 ~ var2, scales = "free")

Add plots where data are not divided by var1, var2:

... + facet\_grid(var1 ~ var2, margins=TRUE)

For >2 variables:

... + facet\_grid(var1 ~ var2 + var3)

Facet labels

... + facet\_grid(var1 ~ var2, labeller = label\_both)

Custom facet labels 1

mtcars$cyl2 <- factor(mtcars$cyl,  
labels = c("alpha", "beta", "sqrt(x, y)"))

... + facet\_grid(. ~ cyl2,  
labeller = label\_parsed)

Custom facet labels 2

... + facet\_grid(. ~ cyl2,   
labeller = label\_bquote(.(x) ^ .(x)))

Layout with highest values on top (like a plot)

... + facet\_grid(var1 ~ var2, as.table = FALSE)

Shrink scales to fit output of statistics, not raw data:

... + facet\_grid(var1 ~ var2, shrink = TRUE)

Drop factor levels not used in the data:

... + facet\_grid(var1 ~ var2, drop = TRUE)

## Colours (\* = defaults)

Color scales for points and lines contain "\_colour\_":

... + scale\_colour\_gradient() \*

All scales below are given in point/line form. Replace \_colour\_ with \_fill\_ if you want to give colours for bars, boxplots, areas, or filled points (pch=21-25). Example:

... + scale\_fill\_gradient() \*

Colour scales for continuous variables

... + scale\_colour\_gradient() \*

... + scale\_colour\_gradient(low="grey80", high="black")

... + scale\_colour\_gradient2(low="green", mid="blue", high="red", midpoint=100)

... + scale\_colour\_gradientn(colours = rainbow(7))

... + scale\_colour\_grey(start=0.2, end=0.8)

... + scale\_colour\_brewer(type = "seq",   
palette = 1)

... + scale\_colour\_brewer(type = "div",   
palette = 1)

Setting min and max limit for scale:

... + scale\_colour\_gradient(limits=c(3, 4))

Set breaks for continuous variables (example)

... + scale\_fill\_gradient(low="grey80", high="black", breaks=c(0,2,5,10,20))

Legend with same size for each category

... + scale\_fill\_gradient(low="grey80", high="black", breaks=c(0,2,5,10,20),  
guide = "legend")

Set breaks and colors for discrete variables

... + scale\_colour\_manual(values = c("red2","darkblue","green3),   
breaks = c("A", "B", "C"))

Setting a separate colour for NA values:

... + scale\_colour\_gradient(na.value = "grey50")

Legends for categorical variables: Points/lines

... + scale\_colour\_hue()

... + scale\_colour\_discrete() (equals "hue")

... + scale\_colour\_brewer(type="qual")

... + scale\_colour\_brewer(palette="Set1")

... + scale\_colour\_manual(values = c("red","blue", "green"))

Brewer's sequential palettes: Blues BuGn BuPu GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples RdPu Reds YlGn YlGnBu YlOrBr YlOrRd

Brewer diverging palettes: BrBG PiYG PRGn PuOr RdBu RdGy RdYlBu RdYlGn Spectral

Brewer qualitative palettes: Set1, Set2, Set3, Dark2, Pastel1, Pastel2, Accent, Paired

## Legends

Remove legend

... + theme(legend.position = "none")

Changing legend title

... + scale\_colour\_hue("New label")

... + scale\_fill\_hue("New label")

... + scale\_colour\_gradient("New label")

... + scale\_fill\_gradient("New label")

Add line break in legend title: use "\n"

... + scale\_colour\_hue("Speed\n(km/hour)")

Changing discrete legend

... + scale\_colour\_manual(values = cols,

breaks = c("4", "6", "8"),

labels = c("four", "six", "eight"))

Change legend for a continuous variable from colourbar to discrete scale

... + scale\_colour\_gradient2(guide = "legend")

Change background colour and edge of legend

... + theme(legend.background =   
element\_rect(fill = "pink", color="red", size=2))

Move legend to bottom, left or top

... + theme(legend.position = "bottom")

Move legend inside main plot, using coordinates between 0 and 1

... + theme(legend.position = c(.2, .8))

Change background color

...panel.background=element\_rect(fill="white")

Increase spacing between lines

... + theme(legend.key.height=unit(3,"line"))

Increase thickness of lines in legend only (not in plot)

... + guides(colour = guide\_legend(override.aes = list(size=3)))

Hide unwanted legend (here: size)

ggplot(data, aes(x=x, y=y, color=datatype, size=1)) + geom\_point() + guides(size=FALSE)

## Axis and title labels

... + labs(x = "X var.", y = "Y var.", title = "Plot title")

Convenience functions for x and y labs

... + xlab("X variable") + ylab("Y variable")

Adjusting position (here: lift title slightly higher up)

... + theme(plot.title = element\_text(vjust = 1.5))

Adjusting size and color (here: blue title, double size):

... + theme(plot.title = element\_text(size = rel(2), color = "blue"))

Adjusting text direction (here: rotate x axis labels):

... + theme(axis.text.x=element\_text(angle=90, hjust=1, vjust=0.2))

Change text size of axis labels:

... + theme(axis.title=element\_text(size=rel(1.3))

Increasing size of numbers/text along one or both axes:

... + theme(axis.text.x=element\_text(size=rel(2))

... + theme(axis.text=element\_text(size=rel(1.3))

Increasing text size of strips in facet plots:

... + theme(strip.text=element\_text(size=rel(1.5))

Increasing text size of legend:

... + theme(legend.title=element\_text(size=rel(2),

legend.text=element\_text(size=rel(1.5))

Text elements in theme

plot.title

axis.title (+ axis.title.x and axis.title.y)

axis.text (+ axis.text.x and axis.text.y)

strip.text (+ strip.text.x and strip.text.y)

## Change tick marks, grid lines etc.

Remove minor tick marks

... + theme(panel.grid.minor = element\_blank())

Change appearance of minor grid lines

... + theme(panel.grid.major = element\_line(colour = "blue", linetype="dashed"))

Remove major grid lines

... + theme(panel.grid.major = element\_blank())

Remove major grid lines in one direction

... + theme(panel.grid.major.x = element\_blank())

Set grid lines (minor\_breaks) and axis values (breaks) at custom values

... + scale\_y\_continuous(minor\_breaks =   
seq(0, 100, 5), breaks = seq(0, 100, 10)))

Set grid lines (minor\_breaks) and axis values (breaks) at custom values for time variables

... + scale\_x\_datetime(minor\_breaks = "days", breaks = "months")

Change date format of time variables (e.g., "31.12")

... + scale\_x\_datetime(labels = date\_format("%d.%m"))

## Change backgrounds

White background (+ black grid lines) except facet strips

... + theme\_bw()

Grey (standard) background (+ white grid lines)

... + theme\_bw()

Background of plot panels

... + theme(panel.background =   
element\_rect(fill = "pink"))

Background of outer parts of plot

... + theme(plot.background =   
element\_rect(fill = "pink"))

Background of facet strips

... + theme(strip.background =   
element\_rect(fill = "pink"))

Edge thickness and colour (here: for panels)

... + theme(panel.background =   
element\_rect(colour = "pink", size=2))

## Tile plots

ggplot(data, aes(x=x,y=y)) + geom\_tile(aes(fill=z))

Change colours:

… + scale\_fill\_gradient(low="grey80", high="black")

To show counts as colours in tile plot, make new data set:

DF <- ddply(data, .(Main\_area, art),   
 summarize, counts = length(Year))

ggplot(DF, aes(y=factor(Main\_area), x=art)) +

geom\_tile(aes(fill=log10(counts)))

## Maps

library(ggmap)

Map only (centred on lon, lat)

mapdata <- get\_map(c(lon=mean(lon), lat=mean(lat)), zoom=7)

map <- ggmap(mapdata)

map

Map with points + labels

map +   
scale\_shape\_identity() + scale\_fill\_identity() +  
geom\_point(data=stations, mapping=aes(lon, lat), size=3, col="brown") +  
geom\_text(data=stations, mapping=aes(lon, lat, label=kode), hjust=-0.1, vjust=0.5, col="brown", size=4)

## "Rose" plots

## Annotating

Add lines (for colors, linetype etc., see "Line plots")

... + geom\_vline(aes(xintercept=10))

... + geom\_hline(aes(yintercept=0.05))

... + geom\_hline(aes(yintercept=seq(0.2,1.2,0.2)))

... + geom\_abline(intercept = 37, slope = -5)

Add text

+ annotate("text", x = 5, y = 30,   
label = "Note this outlier")

Add arrow

+ geom\_segment(aes(x = 5, y = 30, xend = 3.5, yend = 25), arrow = arrow(length = unit(0.5, "cm")))

## Bonus: a little bit of the reshape2 package

From wide to tall data format:

- Include all combinations of *subject* and *time*:

melt(smiths, id = c("subject", "time"),  
measure.vars = c("age", "weight", "height"))

- Only include observed combinations of *subject* and *time*:

melt(smiths, id = c("subject", "time"),  
measure.vars = c("age", "weight", "height"),   
na.rm = TRUE)

From tall to wide data format:

- Returns the original data (one column per variable):

dcast(smithsm, subject + time ~ variable)

- if needed, add value.var:

dcast(smithsm, subject + time ~ variable, value.var = "value")

- As above, but returns matrix with subject+time as rowname:

acast(smithsm, subject + time ~ variable)

- Find mean for each subject/variable over times:

dcast(smithsm, subject + variable ~ .,   
mean, na.rm=TRUE)

- Find length per subject/variable combination

dcast(smithsm, subject + variable ~ ., length)

- Makes instead one variable for each subject:

dcast(smithsm, ... ~ subject)