Journal of
Applied & Computational Mathematics

Wickham, J Appl Computat Math 2016, 5:4
DOI: 10.4172/2168-9679.1000321

Case Report Open Access

# About the ggplot2 Package

**Wickham H***

*Assistant Professor of Statistics at Rice University, Houston, USA*

## Introduction

"ggplot2 is an R package for producing statistical, or data, graphics, but it is unlike most other graphics packages because it has a deep underlying grammar. This grammar, based on the Grammar of Graphics, is composed of a set of independent components that can be composed in many different ways [1-3]. Plots can be built up iteratively and edited later. A carefully chosen set of defaults means that most of the time you can produce a publication-quality graphic in seconds, but if you do have special formatting requirements, a comprehensive theming system makes it easy to do what you want [4].

ggplot2 is designed to work in a layered fashion, starting with a layer showing the raw data then adding layers of annotation and statistical summaries [5,6]."

"ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics."

## Tutorial

#### Sample code for the illustration of ggplot2

*### install & load ggplot library*

install.package("ggplot2")

library("ggplot2")

*### show info about the data*

head(diamonds)

head(mtcars)

*### comparison qplot vs ggplot*

*# qplot histogram*

qplot(clarity, data=diamonds, fill=cut, geom="bar")

*# ggplot histogram -> same output*

ggplot(diamonds, aes(clarity, fill=cut)) + geom_bar() (Figure 1)

### How to use qplot

*# scatterplot*

qplot(wt, mpg, data=mtcars)

*# Transform input data with functions*

qplot(log(wt), mpg - 10, data=mtcars)

*# add aesthetic mapping (hint: how does mapping work)*

qplot(wt, mpg, data=mtcars, color=qsec) (Figure 2)

*# change size of points (hint: color/colour, hint: set aesthetic/mapping)*

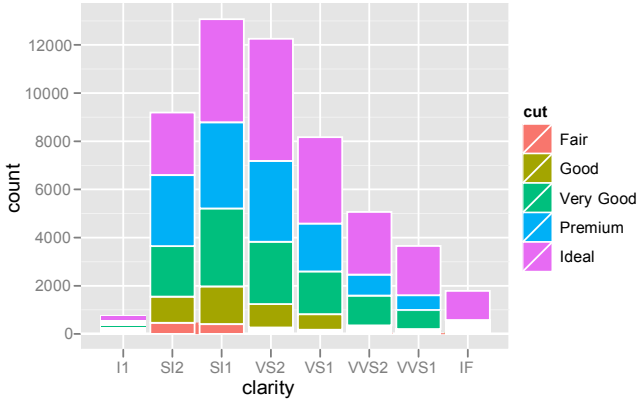


**Figure 1:** Install & load ggplot library.

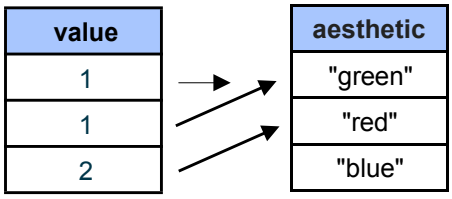


**Figure 2:** Add aesthetic mapping.

qplot(wt, mpg, data=mtcars, color=qsec, size=3)

qplot(wt, mpg, data=mtcars, colour=qsec, size=I(3)) {aesthetics can be set to a constant value instead of mapping}

*# use alpha blending*

qplot(wt, mpg, data=mtcars, alpha=qsec) {values between 0 (transparent) and 1 (opaque)}

*# continuous scale vs. discrete scale*

head(mtcars)

qplot(wt, mpg, data=mtcars, colour=cyl)

levels(mtcars$cyl)

qplot(wt, mpg, data=mtcars, colour=factor(cyl)) (Figure 3)

***Corresponding author:** Wickham H, Assistant Professor of Statistics at Rice University, Houston, USA, Tel: 515 450 8171; E-mail: hadley@rice.rdu

**Received** October 31, 2015; **Accepted** August 25, 2016; **Published** August 31, 2016

**Citation:** Wickham H (2016) About the ggplot2 Package. J Appl Computat Math 5: 321. doi: 10.4172/2168-9679.1000321

*# use different aesthetic mappings*

qplot(wt, mpg, data=mtcars, shape=factor(cyl))

qplot(wt, mpg, data=mtcars, size=qsec)

*# combine mappings (hint: hollow points, geom-concept, legend combination)*

qplot(wt, mpg, data=mtcars, size=qsec, color=factor(carb))

qplot(wt, mpg, data=mtcars, size=qsec, color=factor(carb), shape=I(1))

qplot(wt, mpg, data=mtcars, size=qsec, shape=factor(cyl), geom="point")

qplot(wt, mpg, data=mtcars, size=factor(cyl), geom="point") (Figure 4)

*# bar-plot*

qplot(factor(cyl), data=mtcars, geom="bar")

*# flip plot by 90°*

qplot(factor(cyl), data=mtcars, geom="bar") + coord_flip() {flips the plot after calculation of any summary statistics}

*# difference between fill/color bars*

qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(cyl))

qplot(factor(cyl), data=mtcars, geom="bar", colour=factor(cyl)) (Figure 5)

*# fill by variable*

qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(gear))

*# use different display of bars (stacked, dodged, identity)*

head(diamonds)

qplot(clarity, data=diamonds, geom="bar", fill=cut, position="stack")

qplot(clarity, data=diamonds, geom="bar", fill=cut, position="dodge")

qplot(clarity, data=diamonds, geom="bar", fill=cut, position="fill")

qplot(clarity, data=diamonds, geom="bar", fill=cut, position="identity")

qplot(clarity, data=diamonds, geom="freqpoly", group=cut, colour=cut, position="identity")

qplot(clarity, data=diamonds, geom="freqpoly", group=cut, colour=cut, position="stack") (Figure 6)

*# using pre-calculated tables or weights (hint: usage of ddply in package plyr)*

table(diamonds$cut)

t.table <- ddply(diamonds, c("clarity", "cut"), "nrow")

head(t.table)

qplot(cut, nrow, data=t.table, geom="bar")

qplot(cut, nrow, data=t.table, geom="bar", stat="identity")

qplot(cut, nrow, data=t.table, geom="bar", stat="identity", fill=clarity)

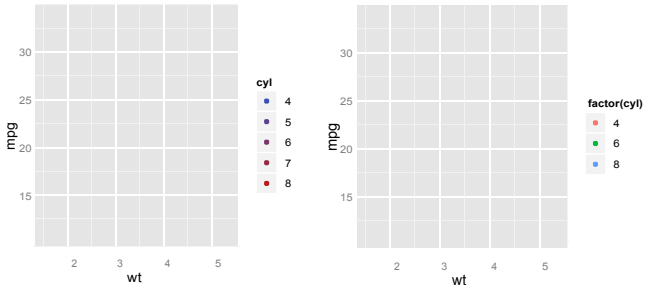


**Figure 3:** Continuous scale vs. discrete scale.

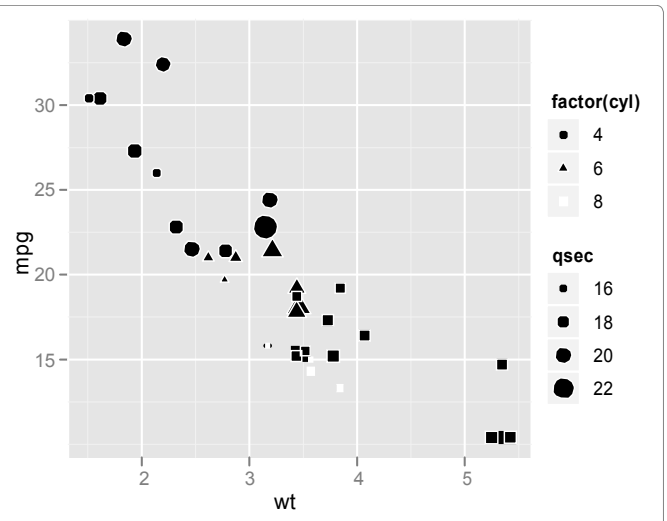


**Figure 4:** Legends are combined if possible (combine mappings).

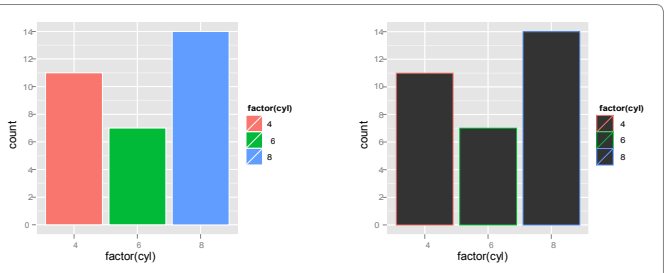


**Figure 5:** Difference between fill/color bars.

qplot(cut, data=diamonds, geom="bar", weight=carat)

qplot(cut, data=diamonds, geom="bar", weight=carat, ylab="carat")

*### Excursion ddply (split data.frame in subframes and apply functions)*

ddply(diamonds, "cut", "nrow")

ddply(diamonds, c("cut", "clarity"), "nrow")

ddply(diamonds, "cut", mean)

ddply(diamonds, "cut", summarise, meanDepth=mean(depth))

ddply(diamonds, "cut", summarise, lower=quantile(depth, 0.25, na.rm=TRUE), median=median(depth, na.rm=TRUE),

upper=quantile(depth, 0.75, na.rm=TRUE))

t.function <- function(x,y){

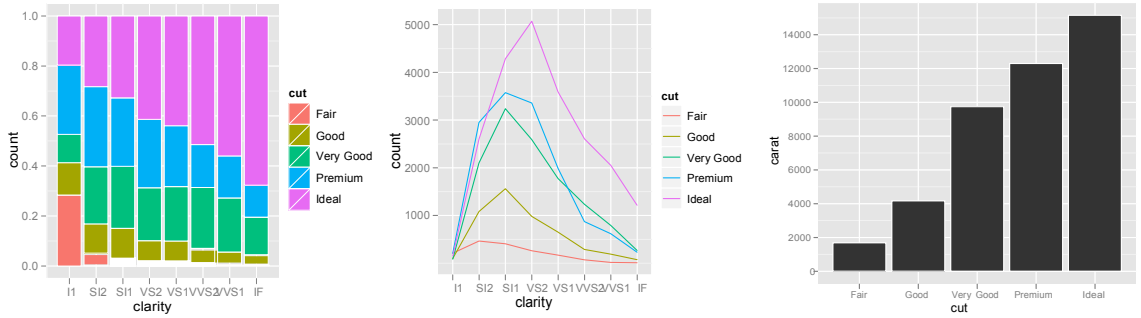


**Figure 6:** Different display of bars.

```
    z=sum(x)/sum(x+y)
    return(z)
    }
    ddply(diamonds, "cut", summarise, custom=t.function(depth, price))
    ddply(diamonds, "cut", summarise, custom=sum(depth)/sum(depth + price))
    ### Back to ggplot
    # histogram
    qplot(carat, data=diamonds, geom="histogram")
    # change binwidth
    qplot(carat, data=diamonds, geom="histogram", binwidth=0.1)
    qplot(carat, data=diamonds, geom="histogram", binwidth=0.01) (Figure 7)
    # use geom to combine plots (hint: order of layers)
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"))
    qplot(wt, mpg, data=mtcars, geom=c("smooth", "point"))
    qplot(wt, mpg, data=mtcars, color=factor(cyl), geom=c("point", "smooth")) (Figure 8)
    # tweeking the smooth plot ("loess"-method: polynomial surface using local fitting)
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"))
    # removing standard error
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"), se=FALSE)
    # making line more or less wiggly (span: 0-1)
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"), span=0.6)
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"), span=1)
    # using linear modelling
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"), method="lm") (Figure 9)
    # using a custom formula for fitting
    Library (splines)
    qplot(wt, mpg, data=mtcars, geom=c("point", "smooth"), method="lm", formula=y ~ ns(x,5)) (Figure 10)
    # illustrate flip versus changing of variable allocation
    qplot(mpg, wt, data=mtcars, facets=cyl~., geom=c("point", "smooth"))
    qplot(mpg, wt, data=mtcars, facets=cyl~., geom=c("point", "smooth")) + coord_flip()
    qplot(wt, mpg, data=mtcars, facets=cyl~., geom=c("point", "smooth")) { flips the plot after calculation of any summary statistics}
    # save plot in variable (hint: data is saved in plot, changes in data do not change plot-data)
    p.tmp <- qplot(factor(cyl), wt, data=mtcars, geom="boxplot")
    p.tmp
    # save mtcars in tmp-var
    t.mtcars <- mtcars
    # change mtcars
    mtcars <- transform(mtcars, wt=wt^2) (Figure 11)
    # draw plot without/with update of plot data
    p.tmp
    p.tmp %+% mtcars
    # reset mtcars
    mtcars <- t.mtcars
    rm(t.mtcars)
    # get information about plot
    summary(p.tmp) (Figure 12)
    # save plot (with data included)
    save(p.tmp, file="temp.rData")
    # save image of plot on disk (hint: svg device must be installed)
    ggsave(file="test.pdf")
    ggsave(file="test.jpeg", dpi=72)
    ggsave(file="test.svg", plot=p.tmp, width=10, height=5)
    ### Going further with ggplot
    # create basic plot (hint: cannot be displayed, no layers yet)
```

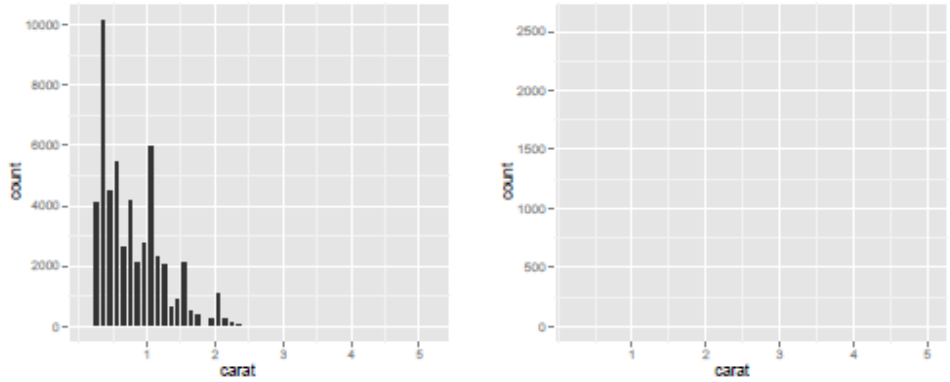


**Figure 7:** Different binwidth changes the picture.

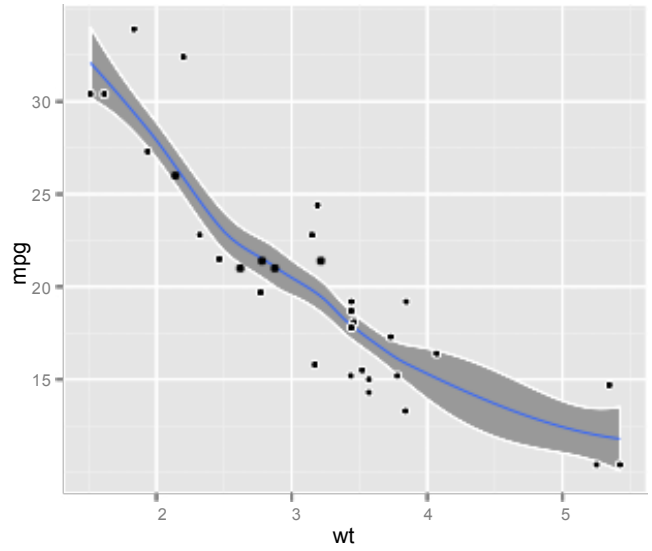


**Figure 8:** Order of layers.

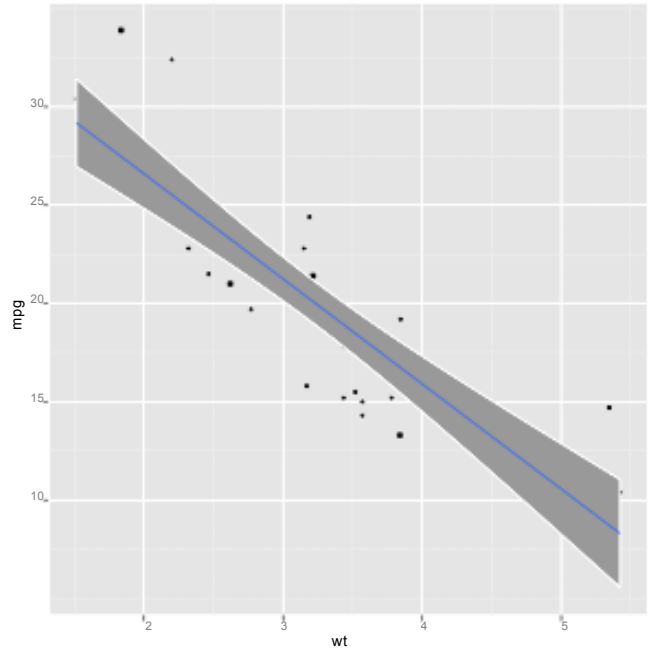


**Figure 9:** Tweeking the smooth plot using linear modelling.

```
p.tmp <- ggplot(mtcars, aes(mpg, wt, colour=factor(cyl)))
p.tmp (Figure 13)
# using additional layers (hint: ggplot draws in layers)
p.tmp + layer(geom="point")
p.tmp + layer(geom="point") + layer(geom="line")
# using shortcuts -> geom_XXX(mapping, data, ..., geom, position)
p.tmp + geom_point()
# using ggplot-syntax with qplot (hint: qplot creates layers automatically)
qplot(mpg, wt, data=mtcars, color=factor(cyl), geom="point") + geom_line()
qplot(mpg, wt, data=mtcars, color=factor(cyl), geom=c("point","line"))
# add an additional layer with different mapping
p.tmp + geom_point()
p.tmp + geom_point() + geom_point(aes(y=disp))
# setting aesthetics instead of mapping
p.tmp + geom_point(color="darkblue")
p.tmp + geom_point(aes(color="darkblue"))
# dealing with overplotting (hollow points, pixel points, alpha[0-1] )
t.df <- data.frame(x=rnorm(2000), y=rnorm(2000))
p.norm <- ggplot(t.df, aes(x,y))
p.norm + geom_point()
p.norm + geom_point(shape=1)
p.norm + geom_point(shape=".")
p.norm + geom_point(colour=alpha("black", 1/2))
p.norm + geom_point(colour=alpha("blue", 1/10)) (Figure 14)
# using facets (hint: bug in margins -> doesn't work)
```

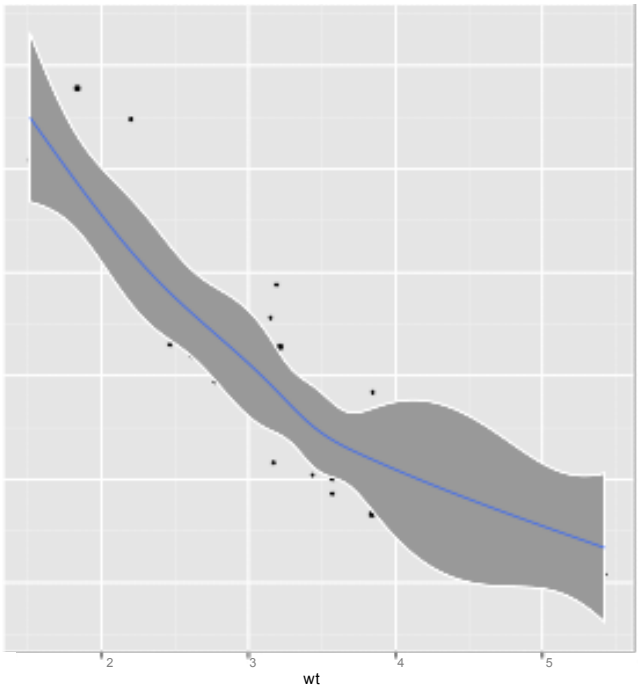


**Figure 10:** Tweeking the smooth plot using library(splines).

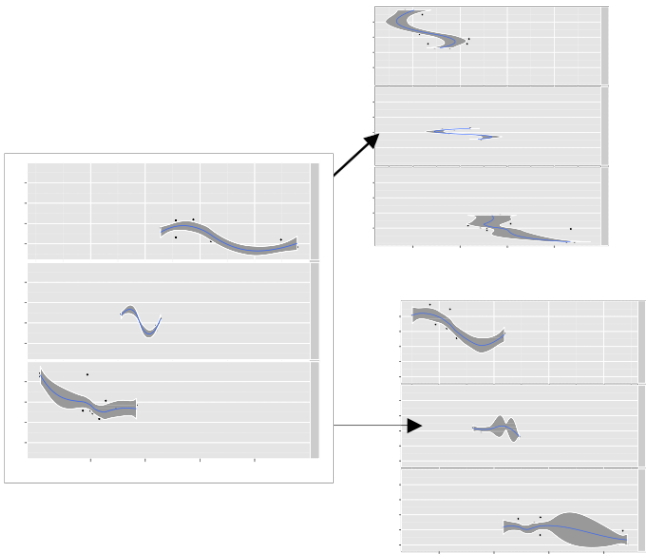

**Figure 11:** Changes in data do not change plot-data.

```
qplot(mpg, wt, data=mtcars, facets=.~cyl, geom="point")
qplot(mpg, wt, data=mtcars, facets=gear~cyl, geom="point")
# facet_wrap/facet_grid
qplot(mpg, wt, data=mtcars, facets=~cyl, geom="point")
p.tmp <- ggplot(mtcars, aes(mpg, wt)) + geom_point()
p.tmp + facet_wrap(~cyl)
p.tmp + facet_wrap(~cyl, ncol=3)
p.tmp + facet_grid(gear~cyl)
```

```
> summary(p.tmp)
data: mpg, cyl, disp, hp, drat, wt, qsec, vs, am,
gear, carb [32x11]
mapping:  x = factor(cyl), y = wt
faceting: facet_grid(. ~ ., FALSE)
.....
geom_boxplot:
stat_boxplot:
position_dodge: (width = NULL, height = NULL)
```

**Figure 12:** Summary (p.tmp).

```
p.tmp + facet_wrap(~cyl+gear)
# controlling scales in facets (default: scales="fixed")
p.tmp + facet_wrap(~cyl, scales="free")
p.tmp + facet_wrap(~cyl, scales="free_x")
p.tmp + facet_wrap(~cyl, scales="fixed")
# contstraint on facet_grid (all rows,columns same scale)
p.tmp + facet_grid(gear~cyl, scales="free_x")
p.tmp + facet_grid(gear~cyl, scales="free", space="free")
# using scales (color palettes, manual colors, matching of colors to values)
p.tmp <- qplot(cut, data=diamonds, geom="bar", fill=cut)
p.tmp
p.tmp + scale_fill_brewer()
p.tmp + scale_fill_brewer(palette="Paired") (Figure 15)
**********RColorBrewer::display.brewer.all()
p.tmp + scale_fill_manual(values=c("#7fc6bc","#083642","#b1df01","#cdef9c","#466b5d")) ].tmp + scale_fill_manual("Color-Matching", c("Fair"="#78ac07", "Good"="#5b99d4",
"Ideal"="#ff9900", "Very Good"="#5d6778", "Premium"="#da0027", "Not used"="#452354")) (Figure 16)
# changing text (directly in qplot/additional shortcut)
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point", xlab="Descr. of x-axis", ylab="Descr. of y-axis", main="Our
Sample Plot")
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") + xlab("x-axis")
# changing name of legend (bug: in labs you must use "colour", "color" doesn't work)
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") + labs(colour="Legend-Name")
# removing legend
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") + scale_color_discrete(legend=FALSE)
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") + opts(legend.position="none")
```

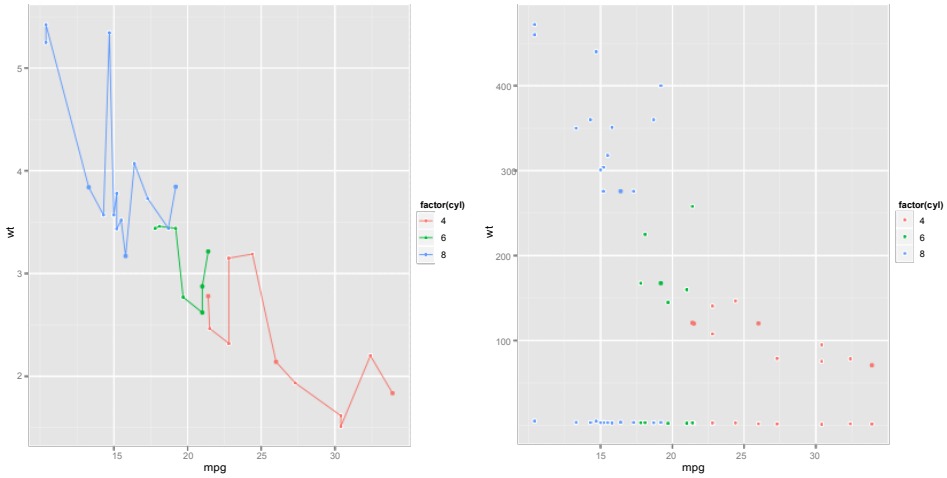


**Figure 13:** Further with ggplot.

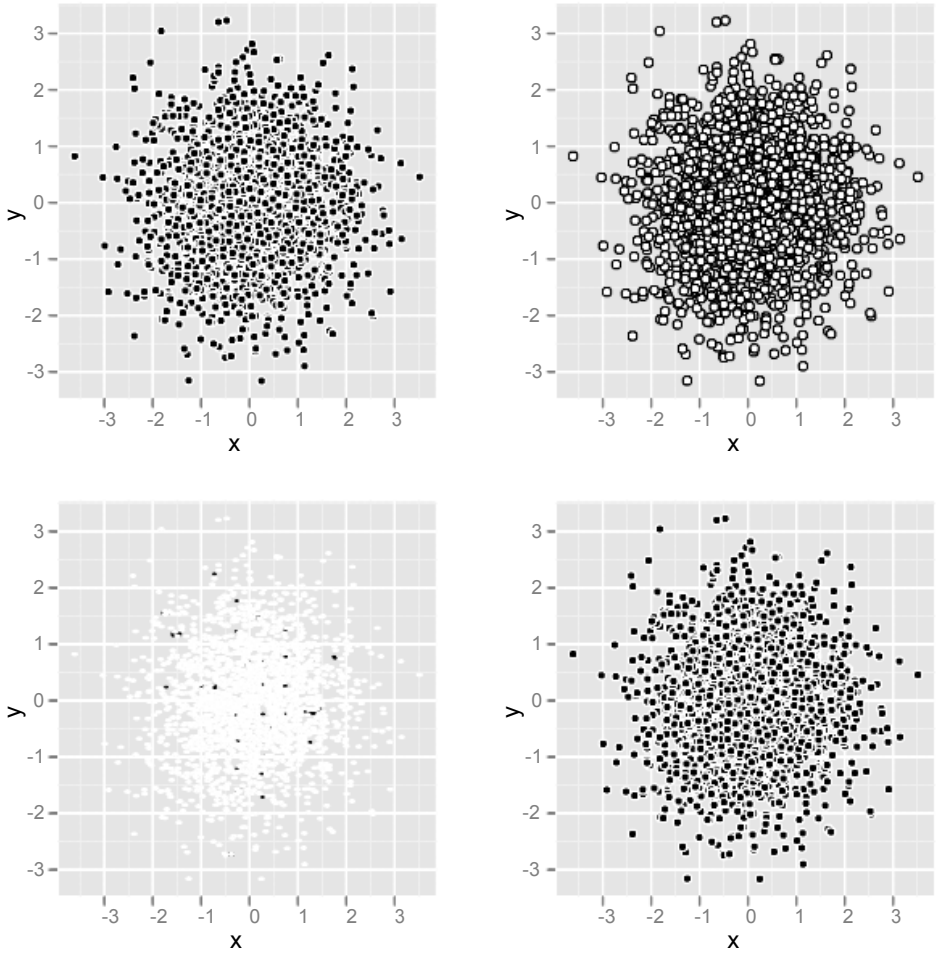


**Figure 14:** Overplotting.

*# moving legend to another place*

qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") + opts(legend.position="left")

*# changing labels on legend*

qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") +

scale_colour_discrete(name="Legend for cyl", breaks=c("4","6","8"), labels=c("four", "six", "eight"))

*# reordering breaks* (*values of legend*)

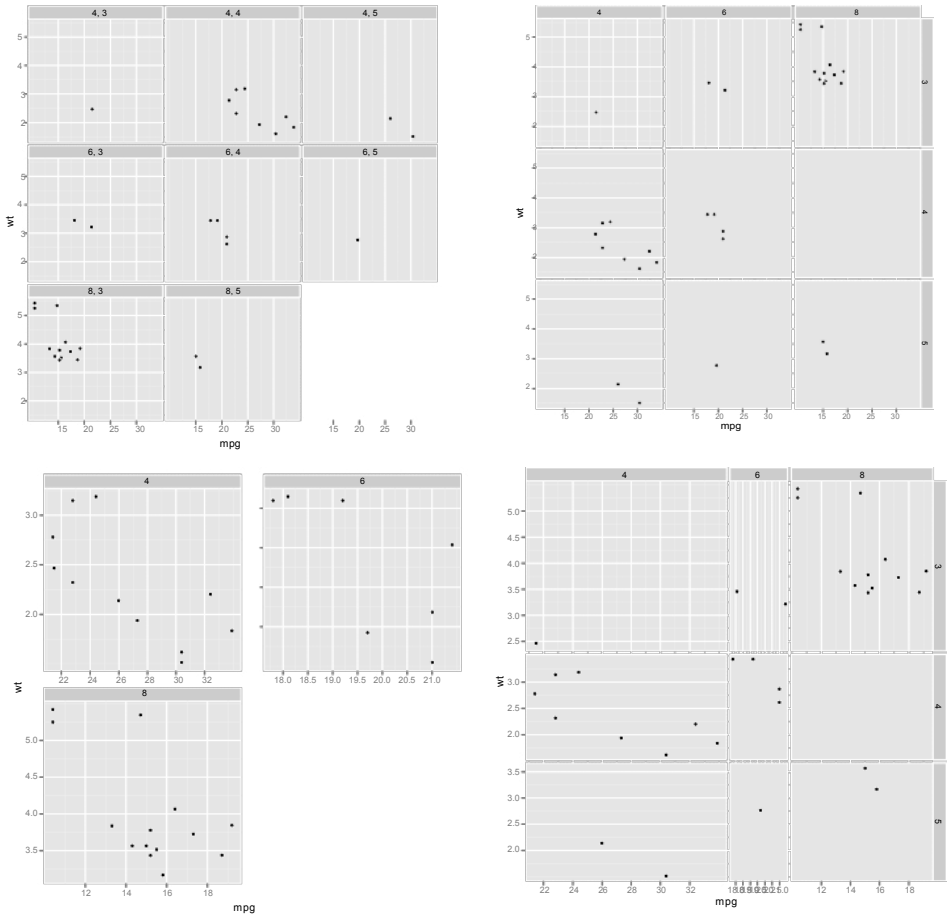


**Figure 15:** GEOM point.

```
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") +
scale_colour_discrete(name="Legend for cyl",breaks=c("8","4","6"))
# dropping factors
mtcars2 <- transform(mtcars, cyl=factor(cyl))
levels(mtcars2$cyl)
qplot(mpg, wt, data=mtcars2, colour=cyl, geom="point") + scale_
colour_discrete(limits=c("4", "8"))
# limits vs zooming in vs breaks
p.tmp <- qplot(wt, mpg, data=mtcars, geom=c("point",
"smooth"), method="lm")
p.tmp
p.tmp + scale_x_continuous(limits=c(15,30))
p.tmp + coord_cartesian(xlim=c(15,30))
p.tmp
p.tmp + scale_x_continuous(breaks=c(15, 18, 27))
p.tmp + scale_x_continuous(breaks=c(15, 18, 27),
labels=c("low", "middle", "high"))
# using transformation
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point")
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") +
scale_y_continuous(trans="log2")
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") +
scale_y_continuous(trans="log2") + scale_x_log10() (Figure 17)
### Themes
# use theme for plot only
qplot(mpg, wt, data=mtcars, geom="point")
qplot(mpg, wt, data=mtcars, geom="point") + theme_bw()
# change font-size for all labels (change base_size)
qplot(mpg, wt, data=mtcars, geom="point") + theme_bw(18)
# change theme for all future plots
theme_set(theme_bw())
# get current theme
theme_get()
# change specific options (hint: "color" does not work in theme_text()
-> use colour)
qplot(mpg, wt, data=mtcars, geom="point", main="THIS IS A
TEST-PLOT")
```

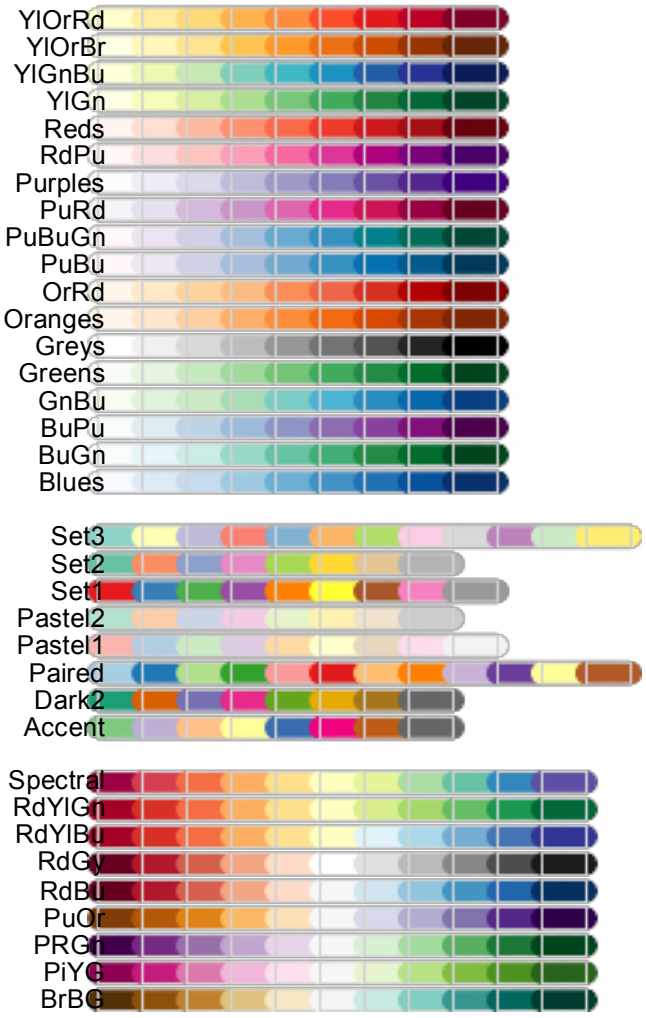


**Figure 16:** RColorBrewer.

qplot(mpg, wt, data=mtcars, geom="point", main="THIS IS A TEST-PLOT") + opts(axis.line=theme_segment(),

plot.title=theme_text(size=20, face="bold", colour="steelblue"), panel.grid.minor=theme_blank(),

panel.background=theme_blank(), panel.grid.major=theme_line(linetype="dotted", colour="lightgrey", size=0.5),

panel.grid.major=theme_blank())

*### create barplot like lattice*

*# use combination of geoms and specific stat for bin calculation*

qplot(x=factor(gear), ymax=..count.., ymin=0, ymax=..count.., label=..count..,

data=mtcars, geom=c("pointrange", "text"), stat="bin", vjust=-0.5,

color=I("blue")) + coord_flip() + theme_bw() (Figure 18)

*### create a pie-chart, radar-chart (hint: not recommended)*

*# map a barchart to a polar coordinate system*

p.tmp <- ggplot(mtcars, aes(x=factor(1), fill=factor(cyl))) + geom_bar(width=1)

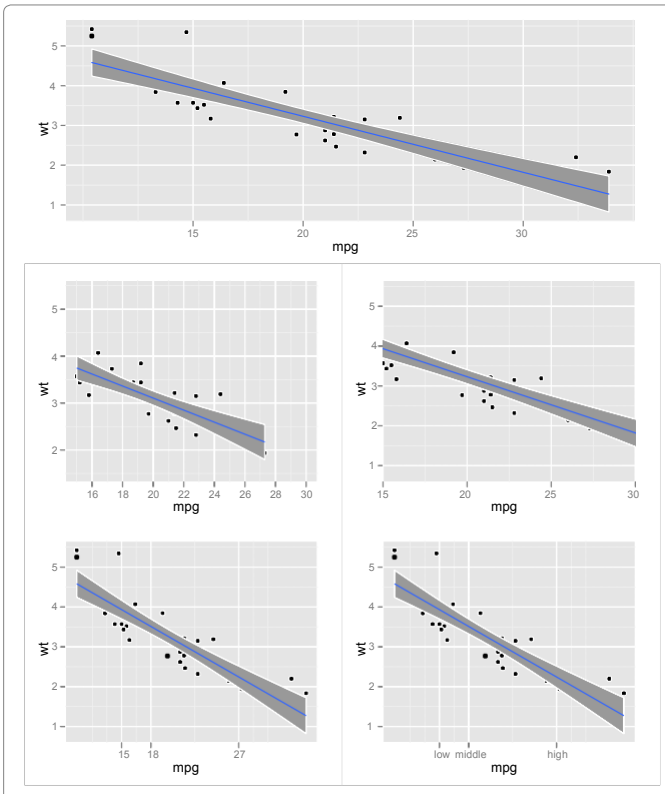


**Figure 17:** Limits vs. zooming in vs. breaks.

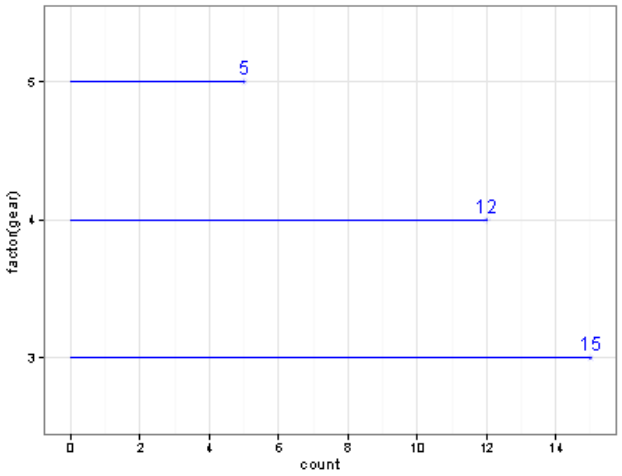


**Figure 18:** Stat for bin calculation.

p.tmp

p.tmp + coord_polar(theta="y")

p.tmp + coord_polar()

ggplot(mtcars, aes(factor(cyl), fill=factor(cyl))) + geom_bar(width=1) + coord_polar (Figure 19)

*### create survival/cumulative incidence plot*

library(survival)

```
head(lung)
# create a kaplan-meier plot with survival package
t.Surv <- Surv(lung$time, lung$status)
t.survfit <- survfit(t.Surv~1, data=lung)
plot(t.survfit, mark.time=TRUE) (Figure 20)
# define custom function to create a survival data.frame
createSurvivalFrame <- function(f.survfit){
# initialise frame variable
f.frame <- NULL
# check if more then one strata
if(length(names(f.survfit$strata))==0){
# create data.frame with data from survfit
f.frame <- data.frame(time=f.survfit$time, n.risk=f.survfit$n.risk, n.event=f.survfit$n.event, n.censor=f.survfit
$n.censor, surv=f.survfit$surv, upper=f.survfit$upper, lower=f.survfit$lower)
# create first two rows (start at 1)
f.start <- data.frame(time=c(0, f.frame$time[1]), n.risk=c(f.survfit$n, f.survfit$n), n.event=c(0,0),
n.censor=c(0,0), surv=c(1,1), upper=c(1,1), lower=c(1,1))
# add first row to dataset
f.frame <- rbind(f.start, f.frame)
# remove temporary data
rm(f.start)
}
else {
# create vector for strata identification
f.strata <- NULL
for(f.i in 1:length(f.survfit$strata)){
# add vector for one strata according to number of rows of strata
f.strata <- c(f.strata, rep(names(f.survfit$strata)[f.i], f.survfit$strata[f.i]))
}
# create data.frame with data from survfit (create column for strata)
f.frame <- data.frame(time=f.survfit$time, n.risk=f.survfit$n.risk, n.event=f.survfit$n.event, n.censor=f.survfit
$n.censor, surv=f.survfit$surv, upper=f.survfit$upper, lower=f.survfit$lower, strata=factor(f.strata))
# remove temporary data
rm(f.strata)
# create first two rows (start at 1) for each strata
for(f.i in 1:length(f.survfit$strata)){
# take only subset for this strata from data
f.subset <- subset(f.frame, strata==names(f.survfit$strata)[f.i])
# create first two rows (time: 0, time of first event)
f.start <- data.frame(time=c(0, f.subset$time[1]), n.risk=rep(f.survfit[f.i]$n, 2), n.event=c(0,0),
n.censor=c(0,0), surv=c(1,1), upper=c(1,1), lower=c(1,1), strata=rep(names(f.survfit$strata)[f.i],
2))
# add first two rows to dataset
f.frame <- rbind(f.start, f.frame)
# remove temporary data
rm(f.start, f.subset)
}
# reorder data
f.frame <- f.frame[order(f.frame$strata, f.frame$time), ]
# rename row.names
rownames(f.frame) <- NULL
}
# return frame
return(f.frame)
}
# define custom function to draw kaplan-meier curve with ggplot
```

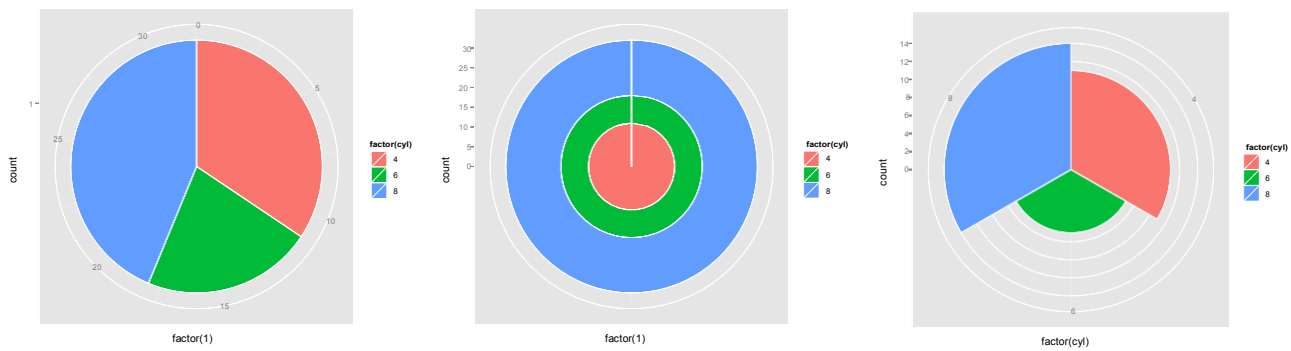


**Figure 19:** pie-chart, radar-chart.

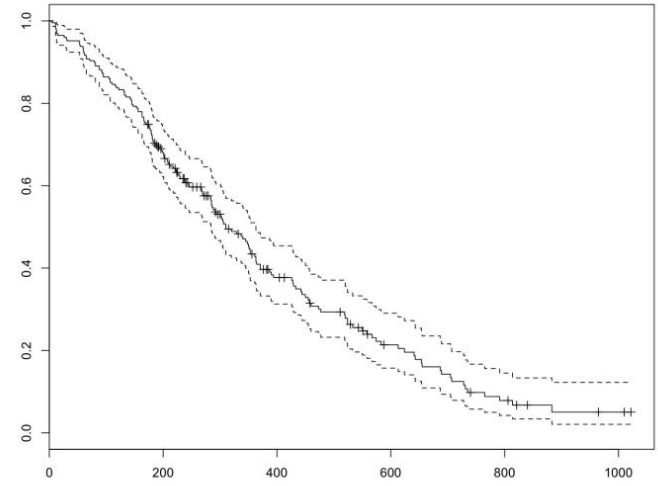


**Figure 20:** Survival/cumulative incidence plot.

```
qplot_survival <- function(f.frame, f.CI="default", f.shape=3){

# use different plotting commands dependig whether or not strata's are given

if("strata" %in% names(f.frame)==FALSE){

# confidence intervals are drawn if not specified otherwise

if(f.CI=="default" | f.CI==TRUE ){

# create plot with 4 layers (first 3 layers only events, last layer only censored)

# hint: censoring data for multiple censoring events at timepoint are overplotted

# (unlike in plot.survfit in survival package)

ggplot(data=f.frame) + geom_step(aes(x=time, y=surv), direction="hv") + geom_step(aes(x=time,

y=upper), directions="hv", linetype=2) + geom_ step(aes(x=time,y=lower), direction="hv", linetype=2) +

geom_point(data=subset(f.frame, n.censor==1), aes(x=time, y=surv), shape=f.shape)

}

else {

# create plot without confidence intervalls

ggplot(data=f.frame) + geom_step(aes(x=time, y=surv), direction="hv") +

geom_point(data=subset(f.frame, n.censor==1), aes(x=time, y=surv), shape=f.shape)

}

}

else {

if(f.CI=="default" | f.CI==FALSE){

# without CI

ggplot(data=f.frame, aes(group=strata, colour=strata)) + geom_ step(aes(x=time, y=surv),

direction="hv") + geom_point(data=subset(f.frame, n.censor==1), aes(x=time, y=surv), shape=f.shape)

}

else {

# with CI (hint: use alpha for CI)

ggplot(data=f.frame, aes(colour=strata, group=strata)) + geom_ step(aes(x=time, y=surv),

direction="hv") + geom_step(aes(x=time, y=upper), directions="hv", linetype=2, alpha=0.5) +

geom_step(aes(x=time,y=lower), direction="hv", linetype=2, alpha=0.5) +

geom_point(data=subset(f.frame, n.censor==1), aes(x=time, y=surv), shape=f.shape)

}

}

}

# create frame from survival class (survfit)

t.survfit <- survfit(t.Surv~1, data=lung)

t.survframe <- createSurvivalFrame(t.survfit)

# create kaplan-meier-plot with ggplot

qplot_survival(t.survframe)

# drawing survival curves with several strata

t.Surv <- Surv(lung$time, lung$status)

t.survfit <- survfit(t.Surv~sex, data=lung)

plot(t.survfit)

# two strata

t.survframe <- createSurvivalFrame(t.survfit)

qplot_survival(t.survframe)

# with CI

qplot_survival(t.survframe, TRUE)

# add ggplot options, use different shape

qplot_survival(t.survframe, TRUE, 1) + theme_bw() + scale_ colour_manual(value=c("green", "steelblue")) +

opts(legend.position="none") (Figure 21)

# multiple strata

t.survfit <- survfit(t.Surv~ph.karno, data=lung)

t.survframe <- createSurvivalFrame(t.survfit)

qplot_survival(t.survframe)

# plot without confidence intervals and with different shape

qplot_survival(t.survframe, FALSE, 20) (Figure 22)
```

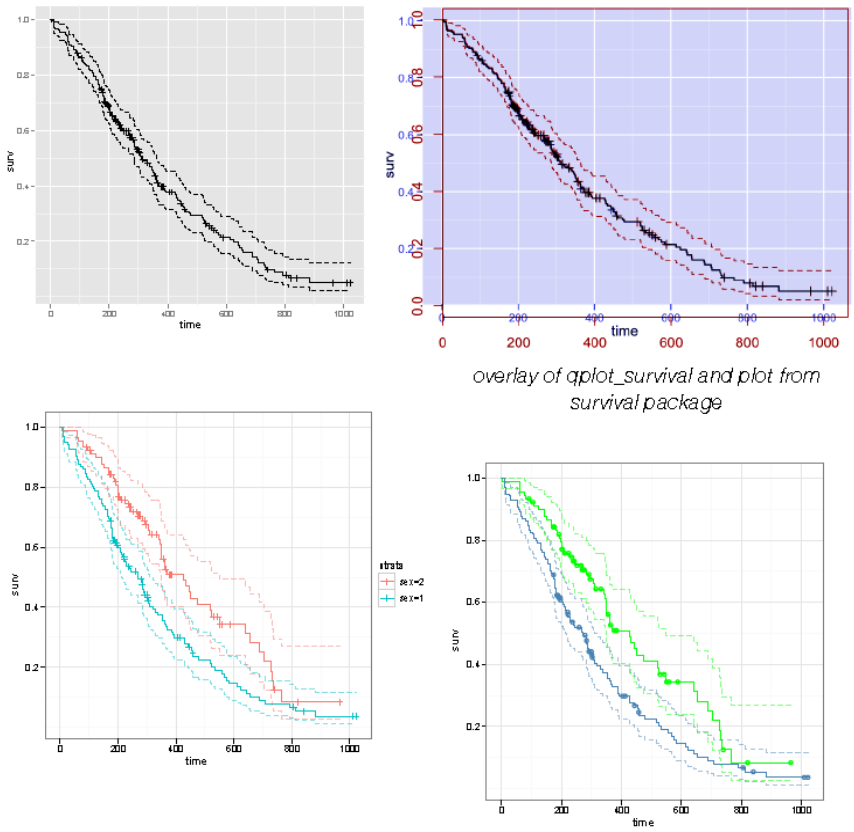


**Figure 21:** Overlay of qplot_survival and plot from survival package.

```
### multiple plots in one graphic
# define function to create multi-plot setup (nrow, ncol) (Figure 23)
vp.setup <- function(x,y){
# create a new layout with grid
grid.newpage()
# define viewports and assign it to grid layout
pushViewport(viewport(layout=grid.layout(x,y)))
}
# define function to easily access layout (row, col)
vp.layout <- function(x,y){
viewport(layout.pos.row=x, layout.pos.col=y)
}
# define graphics
p.a <- qplot(mpg, wt, data=mtcars, geom="point") +
theme_bw()
p.b <- qplot(mpg, wt, data=mtcars, geom="bar",
stat="identity")
p.c <- qplot(mpg, wt, data=mtcars, geom="step")
# setup multi plot with grid
```

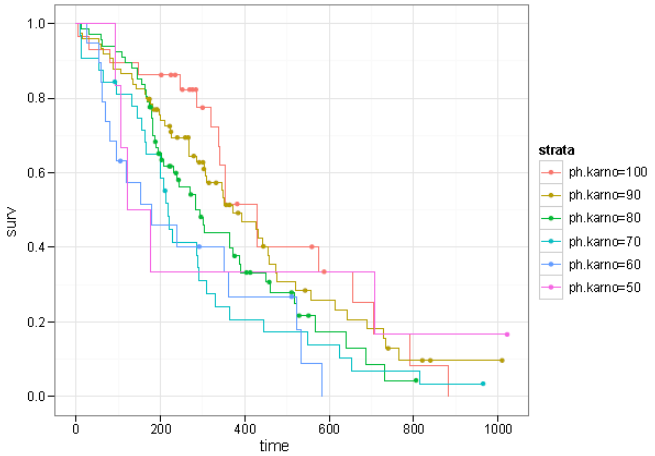


**Figure 22:** Plot without confidence intervals and with different shape.

```
vp.setup(2,2)
# plot graphics into layout
print(p.a, vp=vp.layout(1, 1:2))
print(p.b, vp=vp.layout(2,1))
print(p.c, vp=vp.layout(2,2))
```

Basel Institute for Clinical Epidemiology and Biostatistics

ggplot2

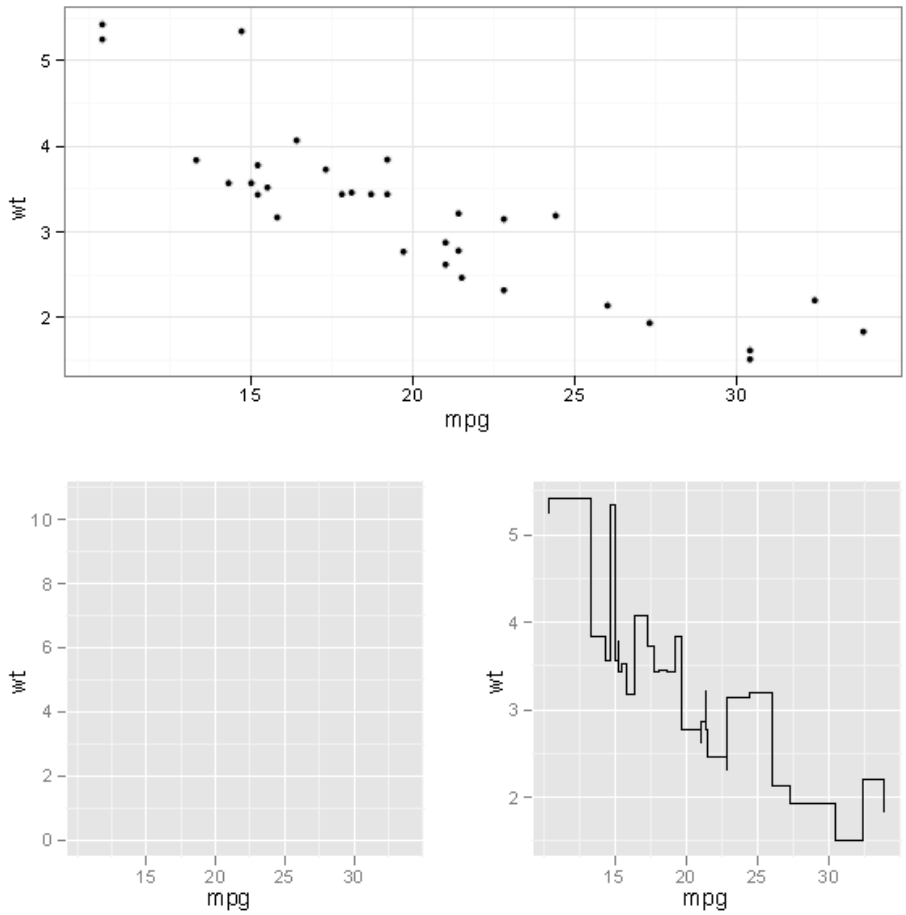


**Figure 23:** Multiple plots in one graphic.

## References

1. Gerke O (2016) Introductory Statistics in a Business Anthropology Program. Anthropol 4: 166.
2. Dubey A, Agarwal SP, Yadav N, Kumar R (2016) Analysis of Caenorhabditis elegans via Bioinformatics Approaches Basis on their Precursors Statistics Values. Next Generat Sequenc & Applic 3: 126.
3. Farooqi MS, Mishra DC, Rai N, Singh DP, Rai A (2016) Genome-Wide Relative Analysis of Codon Usage Bias and Codon Context Pattern in the Bacteria Salinibacter Ruber, Chromohalobacter Salexigens and Rhizobium Etli. Biochem Anal Biochem 5: 257.
4. Grover M (2016) Conservation of Properties of Outer Membranes Protein across Host Genera of Pasteurella multocida Suggests Common Mechanism of Action. Mol Biol 5: 161.
5. Bhati J, Chaduvula PK, Rai A, Gaikwad K, Marla SS (2016) In-Silico Prediction and Functional Analysis of Salt Stress Responsive Genes in Rice (Oryza sativa). J Rice Res 4: 164.
6. Stanton AA (2016) Are Statistics Misleading Sodium Reduction Benefits?. J Med Diagn Meth p: 2713275.