```
library(lattice)

# Daten einlesen
hruch = read.table(file.path(pfadu, "hruch.txt"))
vlax = read.table(file.path(pfadu, "vlax.txt"))
coronal = read.table(file.path(pfadu, "coronal.txt"))
pr = read.table(file.path(pfadu, "preasp.txt"))

dim(hruch)
nrow(hruch)
ncol(hruch)
head(hruch)

bwplot(VOT ~ Alter | Stadt, data = hruch)
bwplot(VOT ~ Stadt | Alter, data = hruch)
bwplot(VOT ~ Stadt | Alter, data = hruch, ylab = "VOT (ms)", xlab = "Stadt",
main = "Meine Daten")
histogram(~ VOT | Alter * Stadt, data = hruch)
densityplot(~ VOT | Alter * Stadt, data = hruch)
d = densityplot(~ VOT | Alter * Stadt, data = hruch, plot.points=F)
b = bwplot(VOT ~ Alter | Stadt, data = hruch)
d2 = update(d, ref=T)

densityplot(~ VOT | Alter * Stadt, data = hruch, plot.points=F)
densityplot(~ VOT | Alter, groups  = Stadt, data = hruch, plot.points=F, auto.key=T)
densityplot(~ VOT, groups = interaction(Alter, Stadt), data = hruch, plot.points=F,
auto.key=T)

# one continuous, dependent variable: bwplot(), densityplot(), histogram()
# densityplot(...groups, plot.points=F, ref=T, auto.key=T)

######################################################## saving trellis graphics and
replotting them
# class(), summary(), plot(g), plot(g[,1]), plot(g[1,1]), plot(t(g))

d = densityplot(~ VOT | Alter * Stadt, data = hruch, plot.points=F)


######################################################## Layout parameters
# bwplot(... layout = c(4,1))
# plot(g, split = c(colnumber, rownumber, ncol, nrow), more =T)
densityplot(~ VOT | Alter * Stadt, data = hruch, plot.points=F, layout = c(4,1))

b = bwplot(VOT ~ Alter | Stadt, data = hruch)
d = densityplot(~ VOT | Alter * Stadt, data = hruch, plot.points=F)
h = histogram(~ VOT | Alter * Stadt, data = hruch)
plot(b, split = c(1, 1, 2, 2), more =T)
plot(h, split = c(1, 2, 2, 2), more =T)
plot(d, split = c(2, 2, 2, 2), more = T)
plot(h[1,1], split = c(2, 1, 2, 2))

######################################################## x-y plot of continuous
variables: xyplot()
```

```
# xyplot(...groups..., auto.key=T)
xyplot(F1 ~ F2 | V * Vpn, data = vlax)
xyplot(-F1 ~ -F2 | Vpn, groups = V, data = vlax, auto.key=T)
xy = xyplot(-F1 ~ -F2 | Vpn, groups = V, data = vlax, auto.key=T)

########################################################## categorical dependent variable:
barchart, dotplot
# tabulating data; proportional data; tab = with(df, table(...); prop = prop.table
(tab, 1:n)
# barchart(), dotplot()
tab = with(coronal, table(Region, Socialclass, Fr))
p = prop.table(tab, 1:2)
barchart(tab, auto.key=T, horizontal=F)
barchart(p, auto.key=T, horizontal=F)
barchart(p, auto.key=T, horizontal=F, layout=c(3,1))
dotplot(p[,,1], auto.key=T, horizontal=F)



###################################################################### colour, lintype
etc
show.settings()
sort(names(trellis.par.get()))

########################################################################## refining bwplot()
# changes can be made to box.rectangle, box.dot, box.umbrella
# which changes? Mostly: col, fill, lty, lwd, cex, pch, transparent
# rectangle
trellis.par.get("box.rectangle")
# the dot (in the middle of the rectangle)
trellis.par.get("box.dot")
# the umbrella
trellis.par.get("box.umbrella")
# the outliers
trellis.par.get("plot.symbol")

bwplot(VOT ~ Alter | Sequenz * Stadt, data = hruch,
par.settings = list(
box.dot = list(col = "slategray"),
box.umbrella = list(col = c("green", "red")),
box.rectangle = list(fill = c("green", "red"), col = c("green", "red")),
plot.symbol = list(col = "black"),
strip.background = list(col = c("white"))
))


bwplot(VOT ~ Alter | Sequenz * Stadt, data = hruch,
par.settings = list(
box.dot = list(col = c("gold", "pink"), cex = 2)
))
```

```
# syntax
bwplot(VOT ~ Alter | Sequenz * Stadt, data = hruch, ylab = "Duration (ms)",
par.settings = list(
box.rectangle = list(fill = c("green", "red")),
box.umbrella = list(col= c("green", "red")),
box.dot = list(col = "black"),
plot.symbol = list(col = "slategray")
))


###################################################################### refining xyplot()
# add grid: xyplot(... type=c("p", "g"))
# change the points: superpose.symbol (syntax as for bwplot())
# see also help(pch), then pchshow()
#
farben = c("gold", "turquoise", "purple", "slategray")
xyplot(-F1 ~ -F2 | Vpn, groups = V, data = vlax, auto.key=T, type = c("p", "g"),
par.settings = list(
superpose.symbol = list(col = farben, pch=c(0, 1, 15, 16), cex = 1.5)
))


###################################################################### refining barchart()
# superpose.polygon = list(...) or simpleTheme()
#
barchart(p, auto.key=T, horizontal=F,
par.settings = list(
superpose.polygon = list(col = c("black", "gold"))
))



###################################################################### refining auto.key()
# auto.key=list(columns = ..., space = ..., text = ...)


###################################################################### The panel function
# some panel functions that are analogous to R functions:
# panel.points, panel.text, panel.abline, panel.curve,

# others
# panel.grid, panel.lmline,

# see panel.superpose

# the above to be called preceding
# panel.barchart, panel.bwplot, panel.densityplot, panel.histogram, panel.xyplot

# e.g.
```

```
pfun = function(...){
panel.grid()
panel.densityplot(...)
}

densityplot(~F1, groups=V, data = vlax, panel=pfun)

pfun = function(x, y){
panel.xyplot(x, y)
if(panel.number()==4)
panel.lmline(x, y)
}
xyplot(F1 ~ F2 | V, data = vlax, panel=pfun)



mfun = function(...){
if(panel.number() == 1)
{
panel.abline(h=40, col=2, lty=2)
panel.text(2, 80, "a")
}
if(panel.number() == 2)
panel.abline(h=60, col=3)
panel.bwplot(...)
}
bwplot(postdur ~ Stadt | Alter, data = hruch, panel=mfun)
```