Grafiken

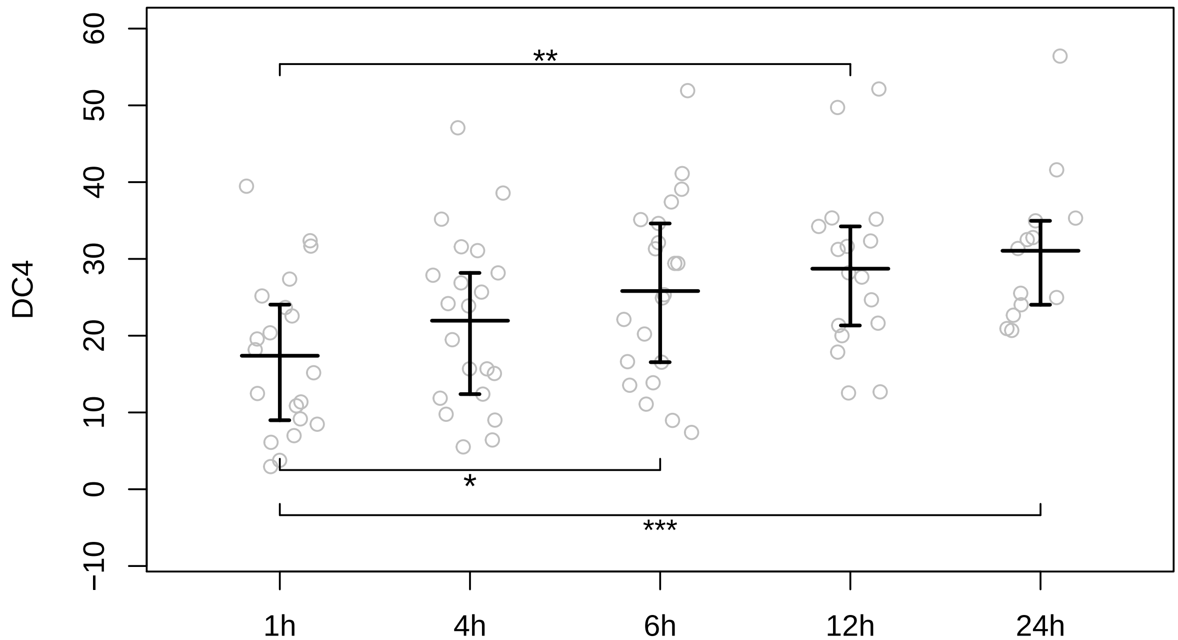
Funktionen

- Theme for lattice-plots (Im Paket stp25output) set_lattice(), set_lattice_ggplot(), set_lattice_bw(), reset_lattice()
- auto_plot Einzelne lattice plots analog wie die Funktion Tabelle()
- Boxplot bwplot2()
- profile_plot()
- plot.bland_altman()
- Hilfsfunktionen wrap_sentence(), stp25plot:::plot.efflist()

Signifikanz-Plot

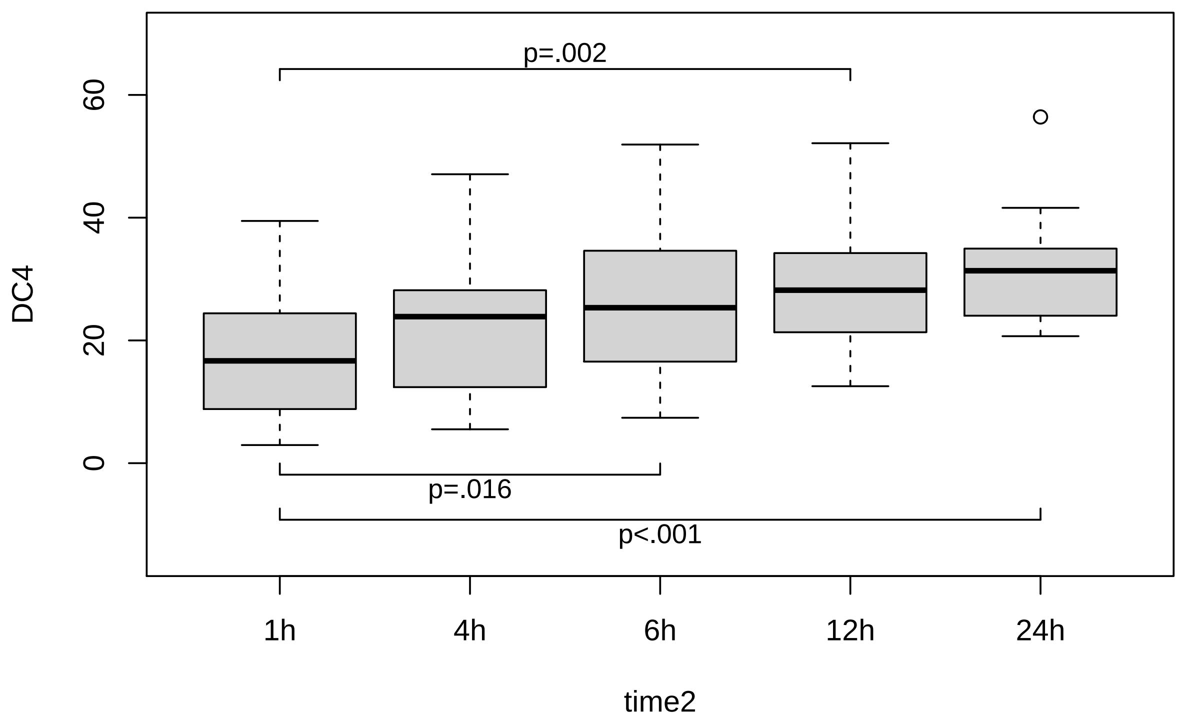
Der Fliegen-Schiss-Plot mein absoluter lieblings Plot!!

```r
#dat1 <- Long(dat, DC4 ~ nmp + time2, value = "DC4")
#fit2 <- lmer(DC4 ~ time2 + (1 | nmp), data = dat1)

fit1 <- lm(DC4 ~ time2, data = dat)
em1 <- emmeans(fit1, list(pairwise ~ time2), adjust = "tukey")
#em2 <- emmeans(fit2, list(pairwise ~ time2))
prism.plots(
  DC4 ~ time2,
  data = dat,
  centerfunc = mean,
  ylim = c(-8, 60)
)
plotSigBars(fit1)
```




```
##   contrast lhs rhs      p.value      p stars
## 5 1h - 24h  1h 24h 0.0007811258 p<.001   ***
## 4 1h - 12h  1h 12h 0.0024623981 p=.002    **
## 3  1h - 6h  1h  6h 0.0163717392 p=.016     *
```

```
boxplot(  DC4 ~ time2,
          data = dat,
          ylim = c(-15, 70))

plotSigBars(fit1, stars=FALSE)
```




```
##   contrast lhs rhs      p.value      p stars
## 5 1h - 24h  1h 24h 0.0007811258 p<.001   ***
## 4 1h - 12h  1h 12h 0.0024623981 p=.002    **
## 3  1h - 6h  1h  6h 0.0163717392 p=.016     *
```

```
#plotSigBars(em1, stars=FALSE)
```

Auto-Plot auto_plot()

Die Funktion klebt lattice- plots zu einer matrix zusammen.

Verwendung: auto_plot(formula, data) oder data %>% auto_plot(var_x, var_y, var_z) Die Funktion kann dabei Formel wie z.B. $a + b + c \sim g$

$a[box] + b[bar] + c[dot] \sim g$

$log(a) + b + c \sim g$

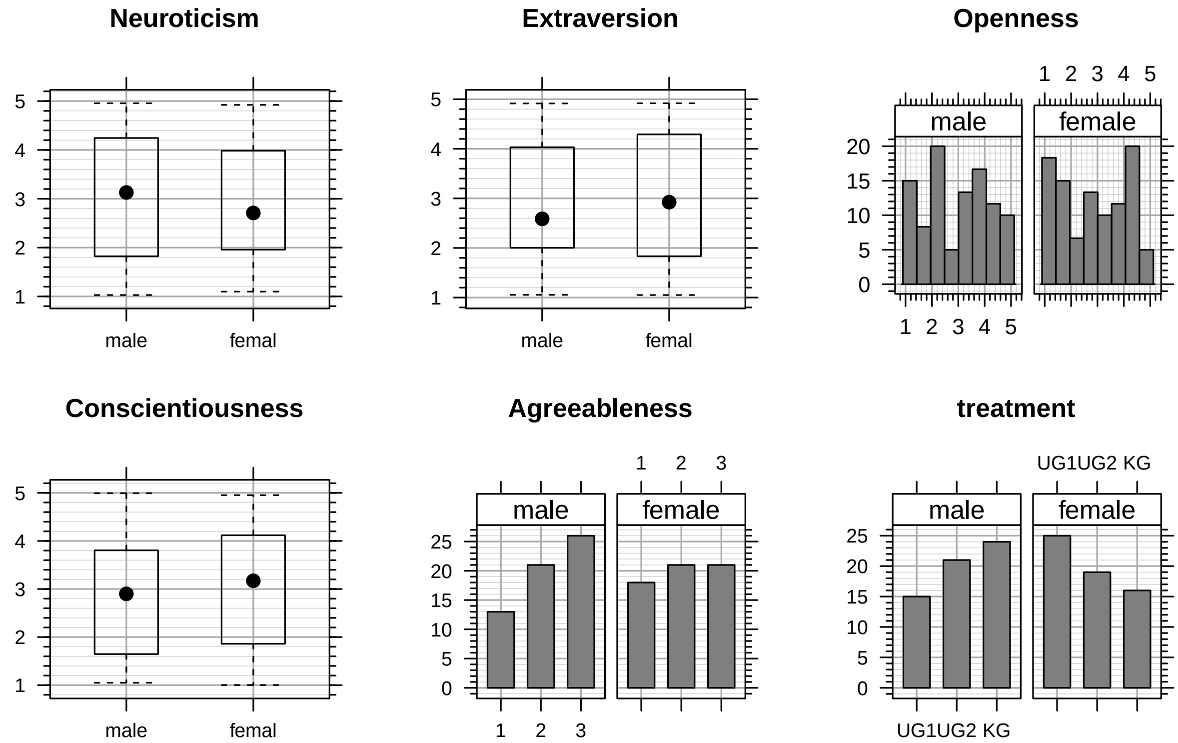
$y \sim a + b + c$

https://www.zahlen-kern.de/editor/

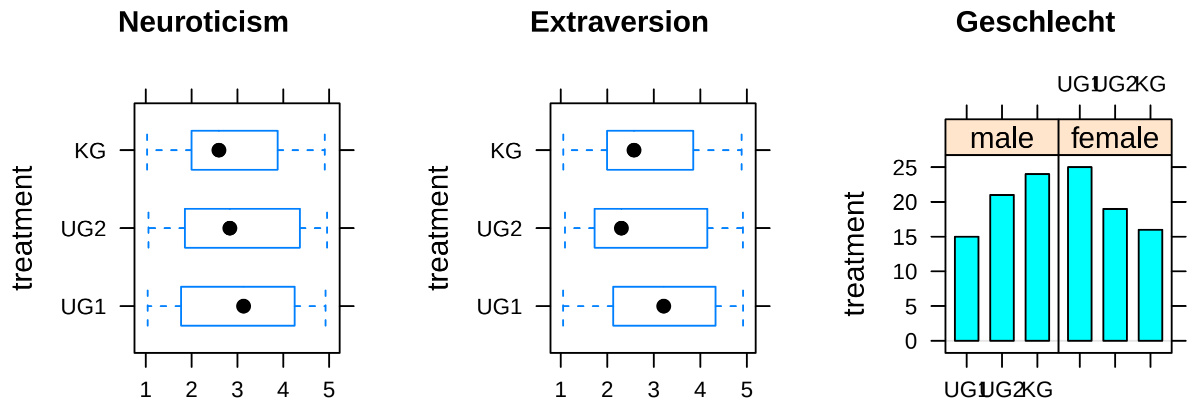
```
DF %>% auto_plot(
  n,
  e[box],
  o[hist],
  g,
  a,
  treatment,
```

```r
  by =  ~ sex,
  par.settings = set_lattice_bw(col = grey.colors(4, start = 0.4, end = 0.9))
)
```

```
##   a    sex Freq
## 1 1   male   13
## 2 2   male   21
## 3 3   male   26
## 4 1 female   18
## 5 2 female   21
## 6 3 female   21
##   treatment    sex Freq
## 1       UG1   male   15
## 2       UG2   male   21
## 3        KG   male   24
## 4       UG1 female   25
## 5       UG2 female   19
## 6        KG female   16
```




```r
reset_lattice()
auto_plot(treatment ~ n + e + sex, DF)
```



set_lattice()

Initialisieren der Lattice - Optionen mit set_lattice(). Im Hintergrund werden die latticeExtra::ggplot2like.opts() aufgerufen und die default Werte in opar und oopt gespeichert um sie mit reset_lattice() zurück seten zu können.

```
my_color <- function(n = 8)  RColorBrewer::brewer.pal(n, "Set2")
my_color_sex <-  function()   RColorBrewer::brewer.pal(8, "Set2")[c(4:3)]
my_color_dark <- function(n = 8)   RColorBrewer::brewer.pal(n, "Dark2")
```

```
reset_lattice()

p1<-barchart(xtabs(~treatment + sex + a,  DF),
             auto.key=list(space="top", columns=3,
                           cex=.7, between=.7 ),
             par.settings= set_lattice())
p2<-barchart(xtabs(~ treatment + sex + a,  DF),
             auto.key=list(space="top", columns=3,
                           cex=.7, between=.7 ),
             par.settings=set_lattice_bw())
p3<-barchart(xtabs(~ treatment + sex + a,  DF),
             auto.key=list(space="top", columns=3,
                           cex=.7, between=.7 ),
             par.settings=set_lattice_ggplot())

grid.arrange(p1, p2, p3, ncol=3)
```

Einbetten von set_lattice() über update()

```
obj <-
  xyplot(
    Sepal.Length + Sepal.Width ~ Petal.Length + Petal.Width,
    iris, type = c("p", "r"),
    jitter.x = TRUE, jitter.y = TRUE, factor = 5,
    auto.key = list(
      cex.title = 1.2,
      title = "Expected Tau",
      text = c("30 ms", "80 ms", "130 ms", "180 ms"),
      space = "top" # lines = TRUE, rectangles = TRUE
    ))
```

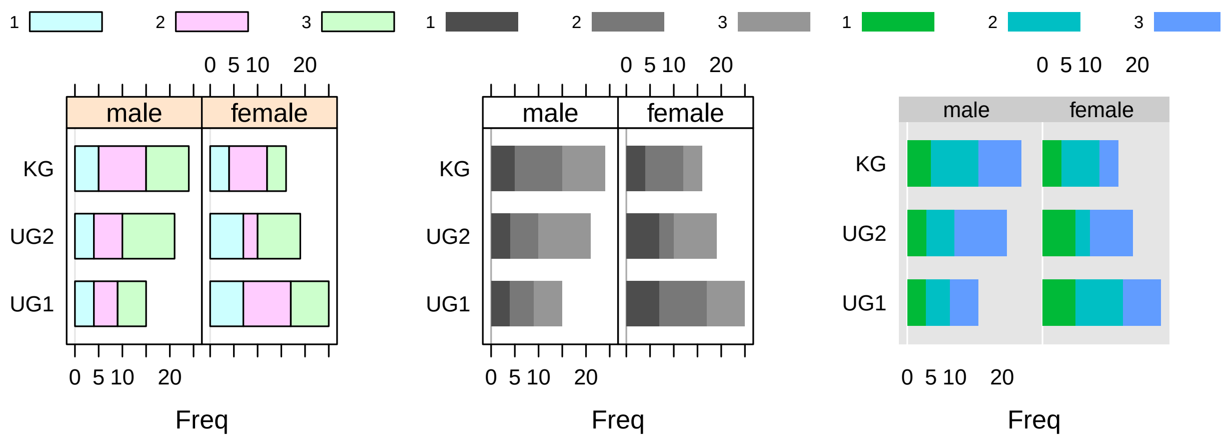


Figure 1: Plot mit grid.arrange - hier muss das Theme mit par.settings= set_lattice() uebergeben werden

```
obj <- update(obj,
              legend = list(
                right =
                  list(fun = "draw.colorkey",
                       args = list(list(at = 0:100)))))

p1 <- update(obj, par.settings = custom.theme( ))
p2 <- update(obj, par.settings = set_lattice(theEconomist.theme()))
p3 <- update(obj, par.settings = set_lattice_bw(), axis = axis.grid)

grid.arrange(p1, p2, p3, ncol = 3)
```

bwplot2

Lattice bwplot mit groups. Ist eine erweiterung von lattice::bwplot. Die Funktion arbeitet mit panel.superpose.

```
p1 <- bwplot2(
  yield ~ site,
  data = barley, groups = year, main="bwplot2()", par.settings = set_lattice_bw(),
  auto.key = list(points = FALSE, rectangles = TRUE, space = "right")

)

p2 <-
  bwplot(
    yield ~ site,
    barley,groups = year, main="panel.superpose", par.settings = set_lattice_bw(),
    auto.key = list(points = FALSE, rectangles = TRUE, space = "right"),
    box.width = 1 / 4,
    panel = function(x, y, groups, subscripts, ...) {
      xx <-
        as.numeric(x) + scale(as.numeric(groups), scale = FALSE)/(nlevels(groups)+1)
      panel.superpose(
```

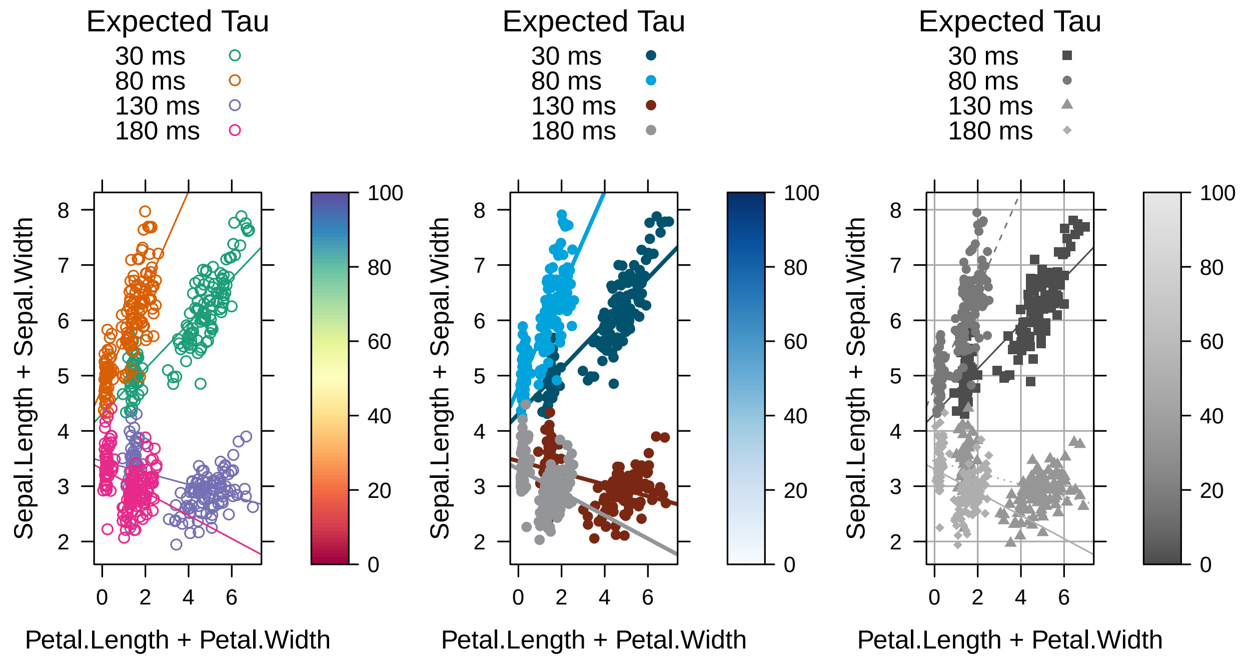


Figure 2: Plot mit grid.arrange und update

```
        xx,  y,  ...,
        panel.groups = panel.bwplot,
        groups = groups,
        subscripts = subscripts
      )
    }
  )


grid.arrange(p1, p2)
```

```
## Warning in (function (x, y, ..., group.number, nlevels, space_between) :
##
## Eine eine Variable muss ein Factor sein die Ander muss Numeric sein!
## x= numeric und y = numeric

## Warning in (function (x, y, ..., group.number, nlevels, space_between) :
##
## Eine eine Variable muss ein Factor sein die Ander muss Numeric sein!
## x= numeric und y = numeric
```

```
set_lattice_bw(col=c("gray80", "gray90"))

bwplot(yield ~ site, data = barley, groups=year,
       pch = "|", box.width = 1/3,
       auto.key = list(points = FALSE, rectangles = TRUE, space = "right"),
       panel = panel.superpose,
       panel.groups = function(x, y, ..., group.number) {
```

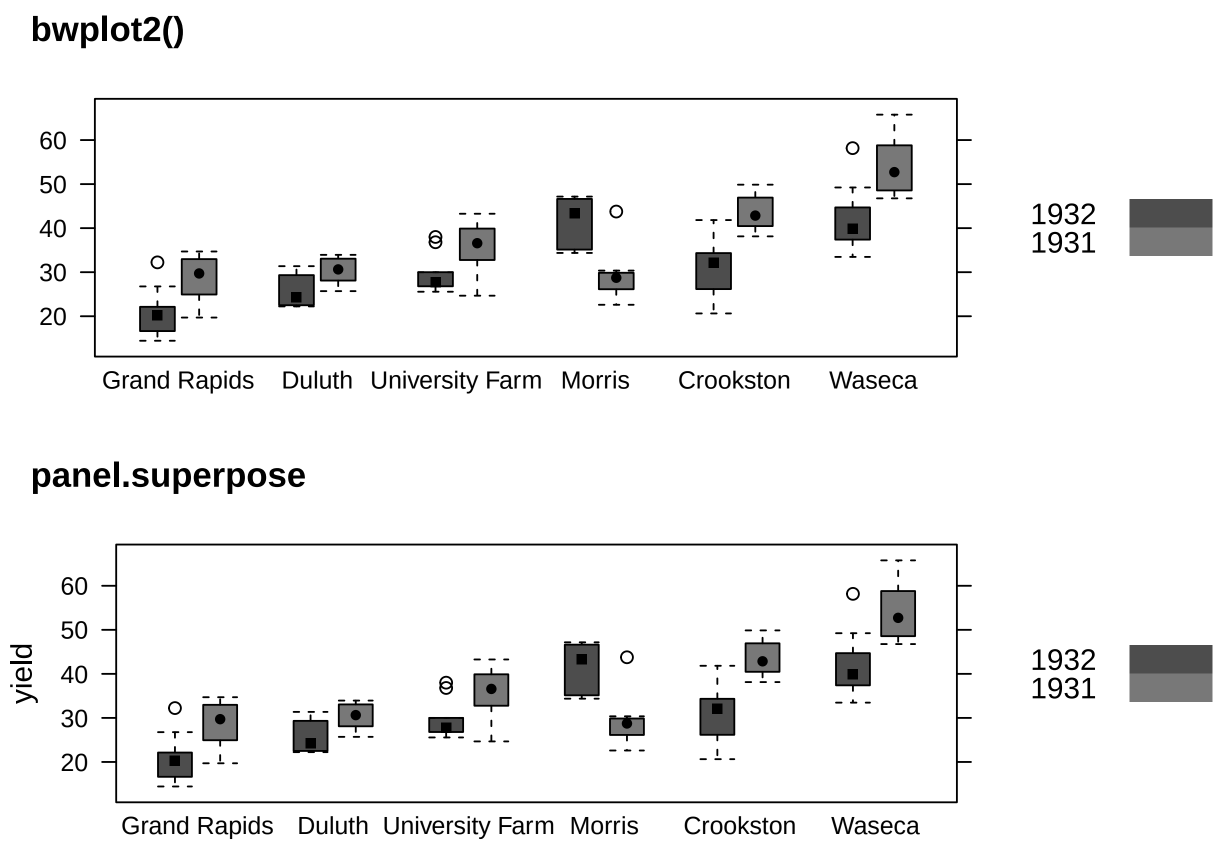


Figure 3: Boxplot mit bwplot2() und panel.superpose()

```
        panel.bwplot(x + (group.number-1.5)/3, y, ...)
        mean.values <- tapply(y, x, mean)
        panel.points(x + (group.number-1.5)/3, mean.values[x], pch=17)
}


)
```

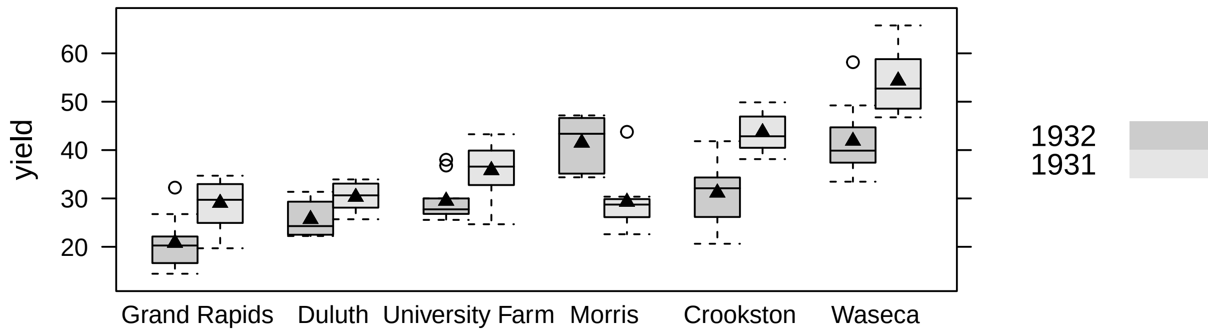


Figure 4: Boxplot mit panel.bwplot() und panel.superpose()

```
  bwplot(
    yield ~ site,
    barley, groups = year, main="panel.superpose", par.settings = set_lattice_bw(),
    auto.key = list( points = FALSE, rectangles = TRUE, space = "right"),
    box.width = 1 / 4,
    panel = function(x, y, groups, subscripts, ...) {
      xx <-
        as.numeric(x) + scale(as.numeric(groups), scale = FALSE) /
        (nlevels(groups)+1)
      panel.superpose(
        xx,  y,  ..., panel.groups = panel.mean,
        groups = groups, subscripts = subscripts
      )
     panel.grid(h = -1, v = 0)
      # panel.stripplot(x, y, ..., jitter.data = TRUE,
      #                 groups = groups, subscripts = subscripts)
      # panel.superpose(x, y, ..., panel.groups = panel.average,
      #                 groups = groups, subscripts = subscripts)
      # panel.points(x, y, ..., panel.groups = panel.average,
      #              groups = groups, subscripts = subscripts)
    }
  )
```

Forest

forest_plot() gestohlen von survminer::ggforest()

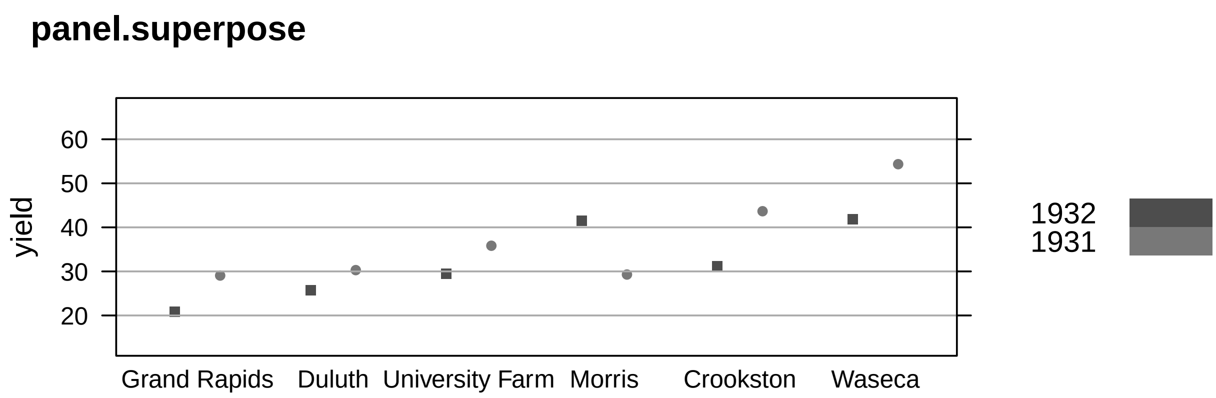
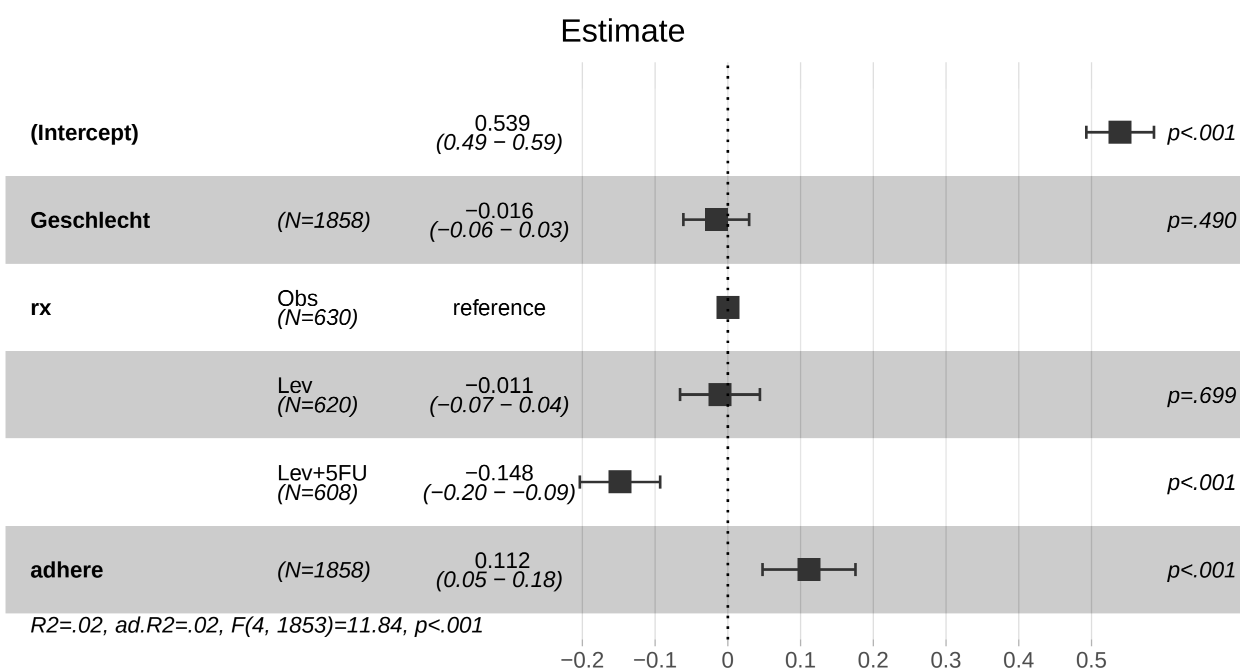


Figure 5: Mittelwerte mit einer Variante von panel.superpose()

```r
fit1 <- lm(status ~ sex + rx + adhere,
           data = colon)
forest_plot(fit1)
```

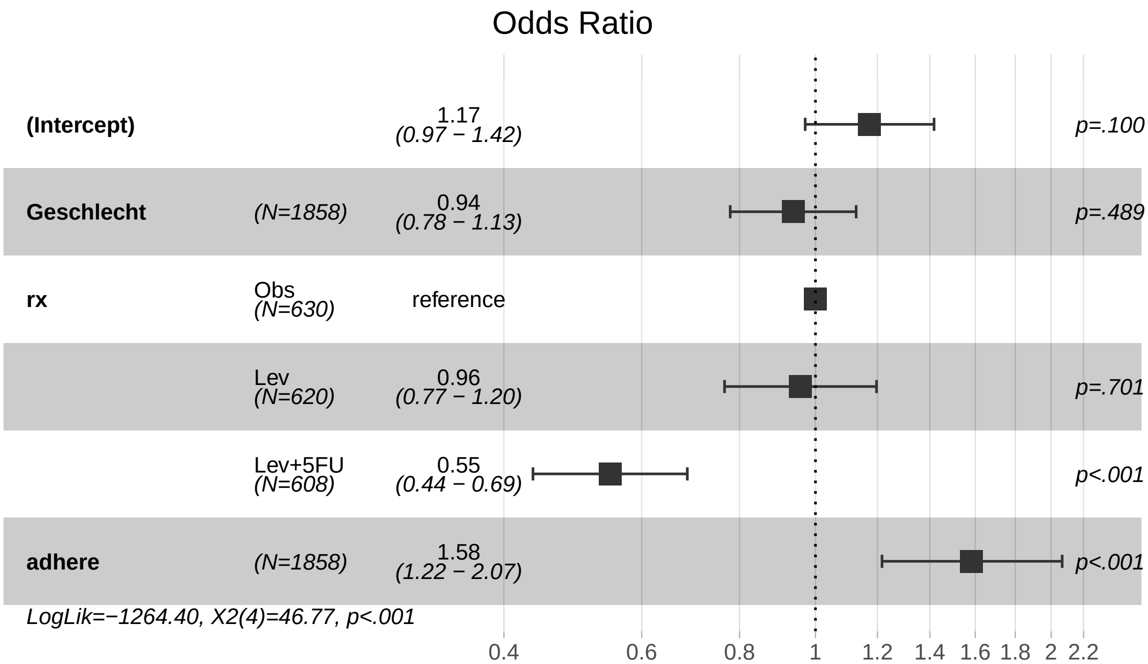
```
##       status          sex           rx       adhere
##     "status" "Geschlecht"         "rx"     "adhere"
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```



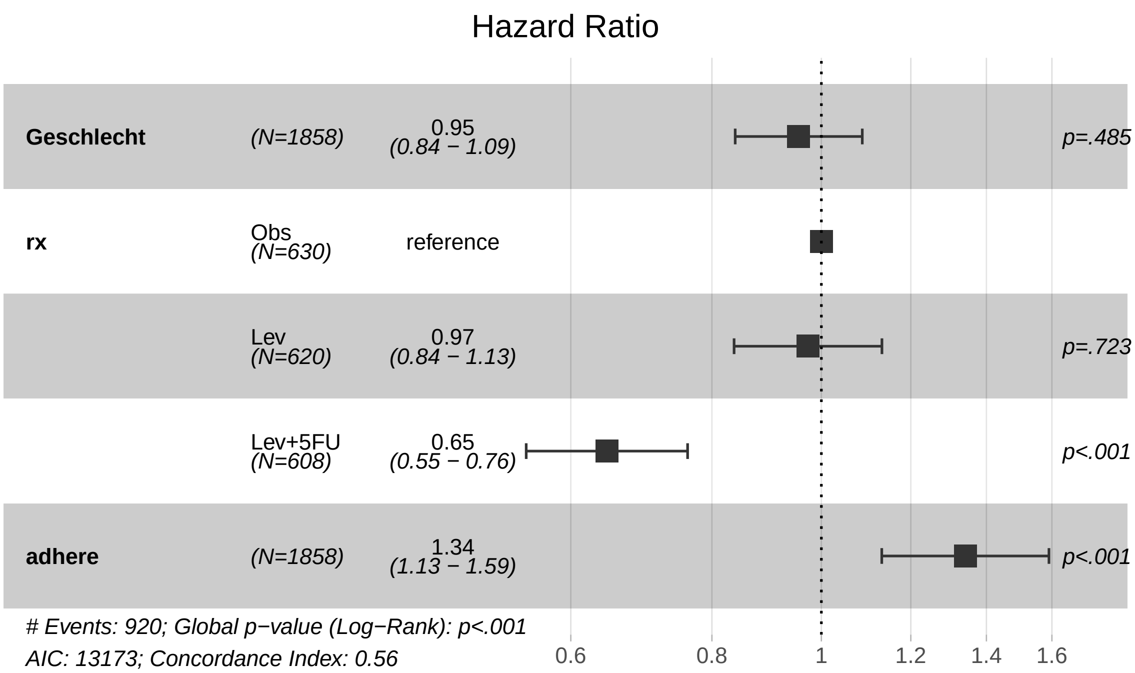

```r
fit2 <- glm(status ~ sex + rx + adhere,
            data = colon, family = binomial())

forest_plot(fit2)
```

```
## Waiting for profiling to be done...
##       status          sex           rx       adhere
##     "status" "Geschlecht"         "rx"     "adhere"
## Warning: Removed 1 rows containing missing values (geom_text).
```



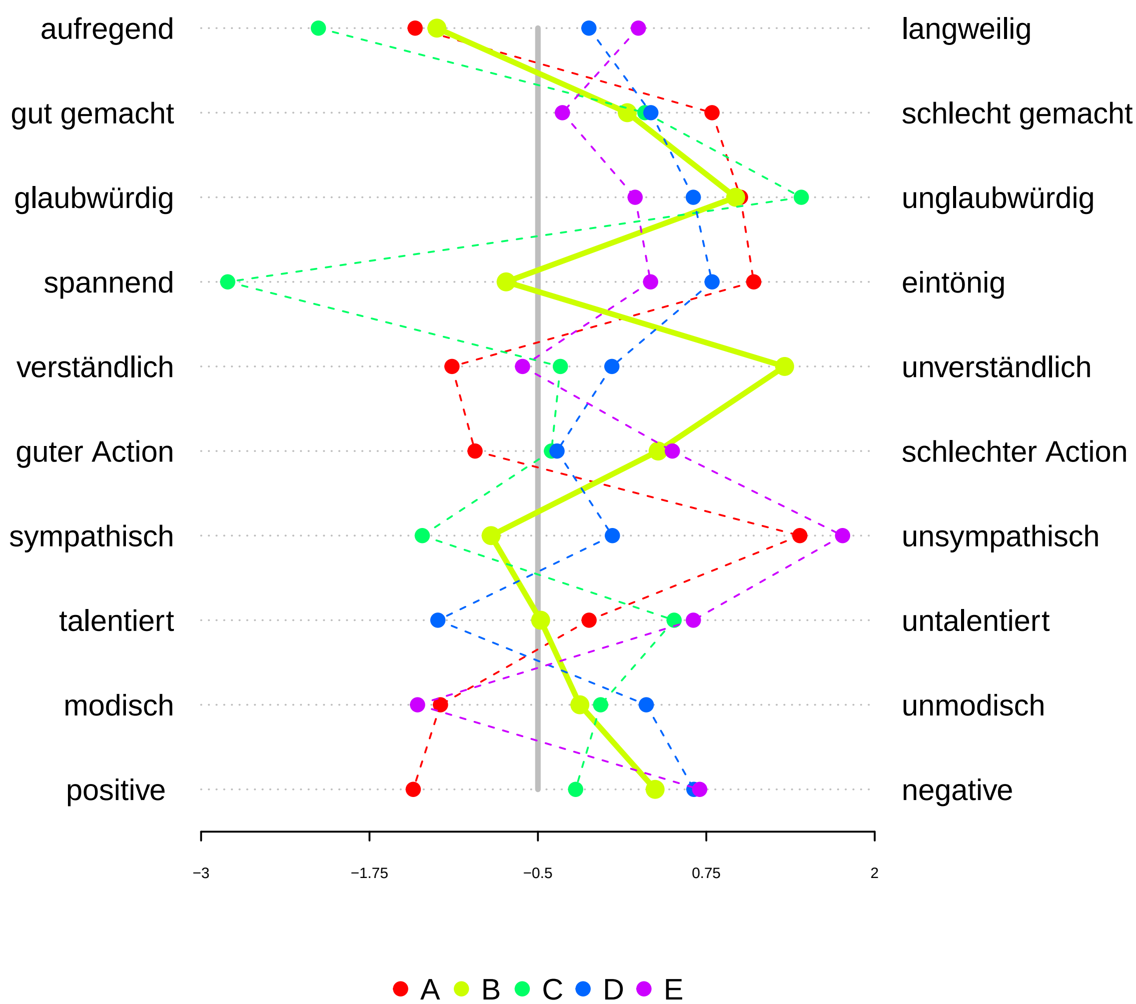

```
fit3 <- coxph(Surv(time, status) ~ sex + rx + adhere,
              data = colon)

forest_plot(fit3, colon)
```

```
##       status          sex           rx       adhere
##     "status" "Geschlecht"         "rx"     "adhere"
## Warning: Removed 1 rows containing missing values (geom_text).
```



profile_plot

```
profile_plot(x,
             highlight.col = 2,
             legend.n.col = 5)
```



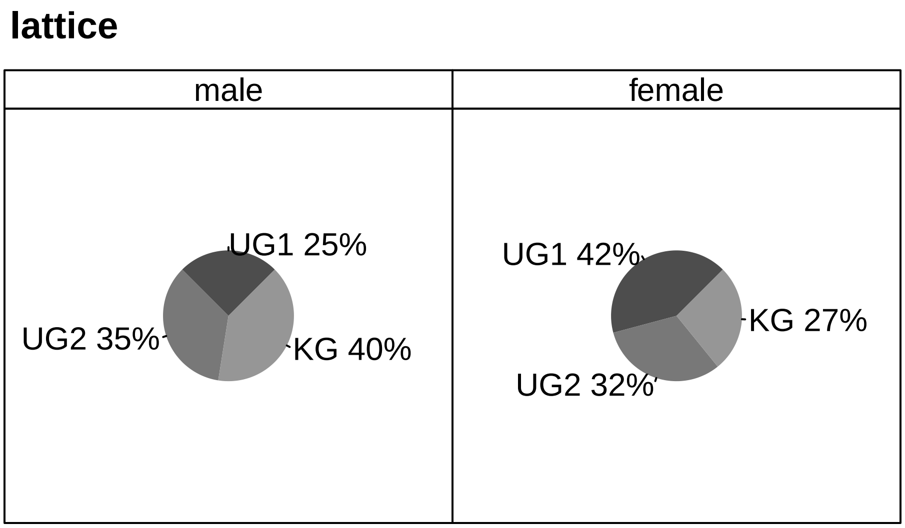

```
## [1] TRUE
```

Tortendiagramme

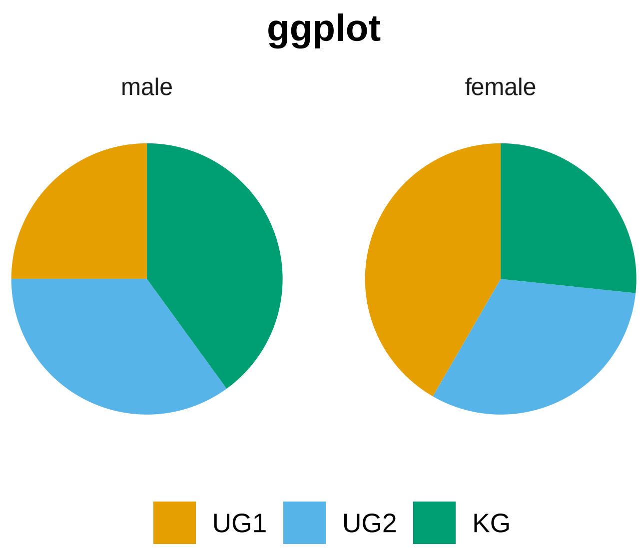
```
print(torte(~treatment+sex, DF, init.angle=45, main="lattice"))
```

```
## Loading required package: gridBase
```

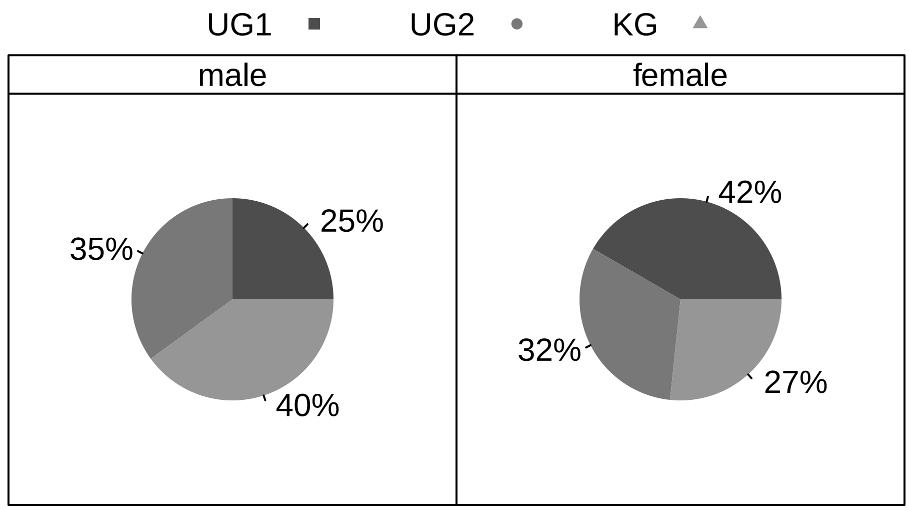
```
## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.

## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.
```




```
gtorte(~treatment+sex, DF, init.angle=45, main="ggplot")
```



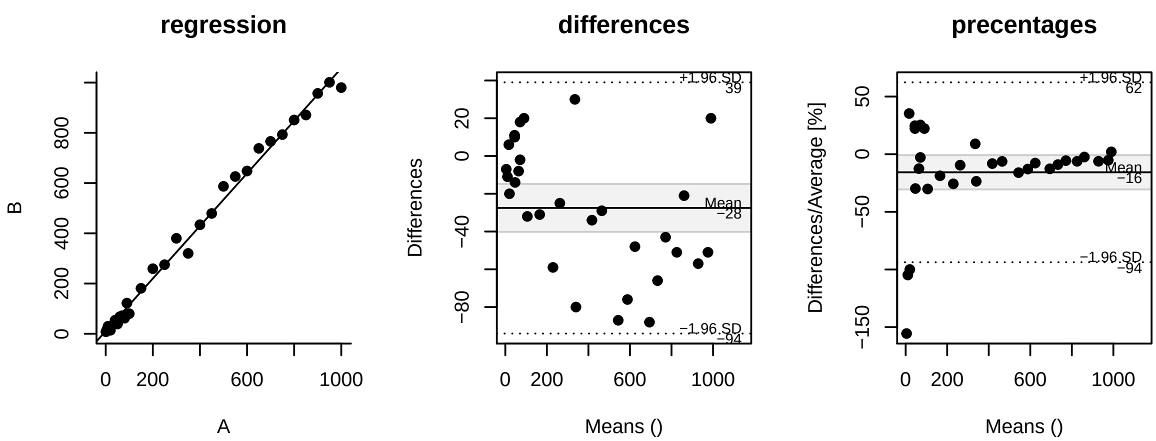

```
#  Geht nicht problemlos in Markdown
 tab <- as.data.frame(xtabs( ~ treatment + sex, DF))
# par(new = TRUE)
  stp25plot::piechart(~Freq|sex, tab, groups= treatment, auto.key=list(columns=3))
```



MetComp_BAP

Tukey Mean Difference oder auch Bland Altman Metode

```
x<- MetComp_BAP(~A+B, DF2)
plot(x)
```



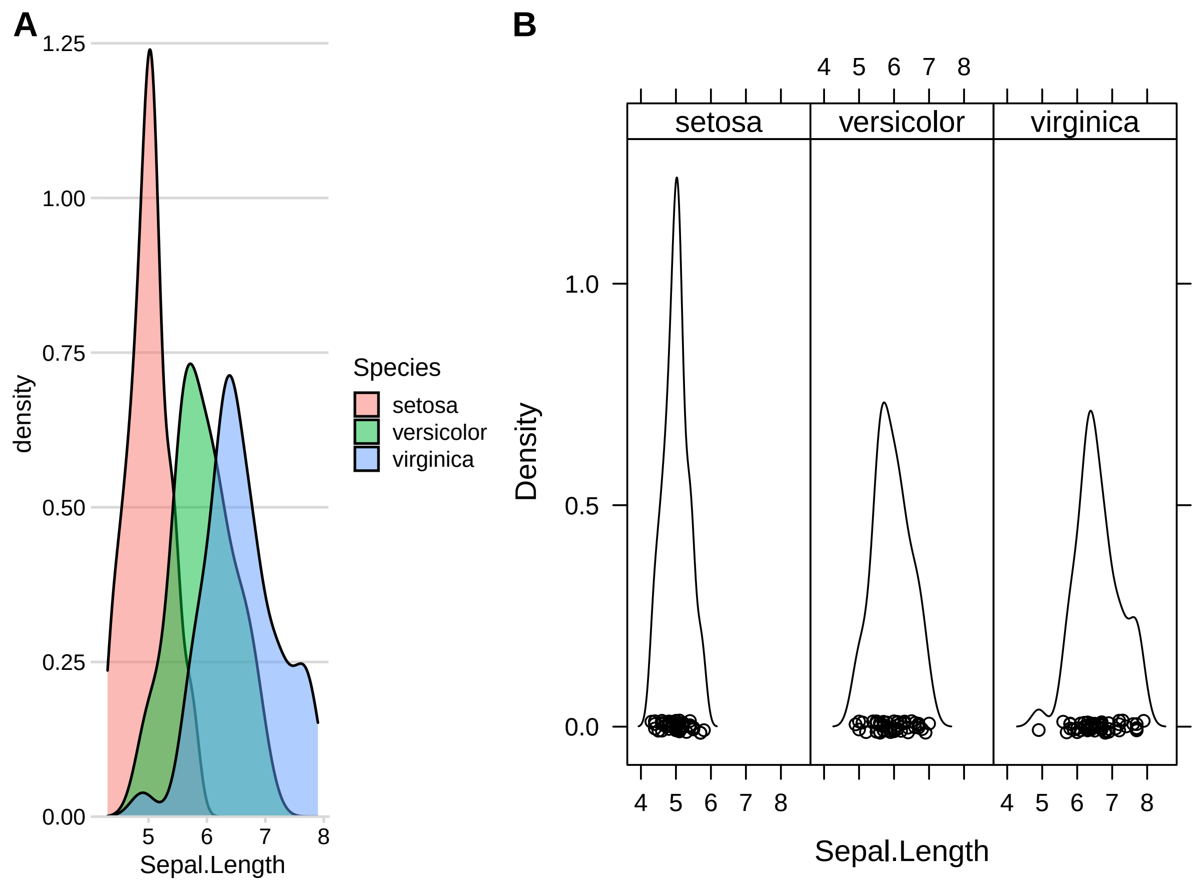
cowplot

Zusammen mixen von unterschiedlichen Grafik-Typen.

The cowplot package is a simple add-on to ggplot. https://wilkelab.org/cowplot/articles/index.html

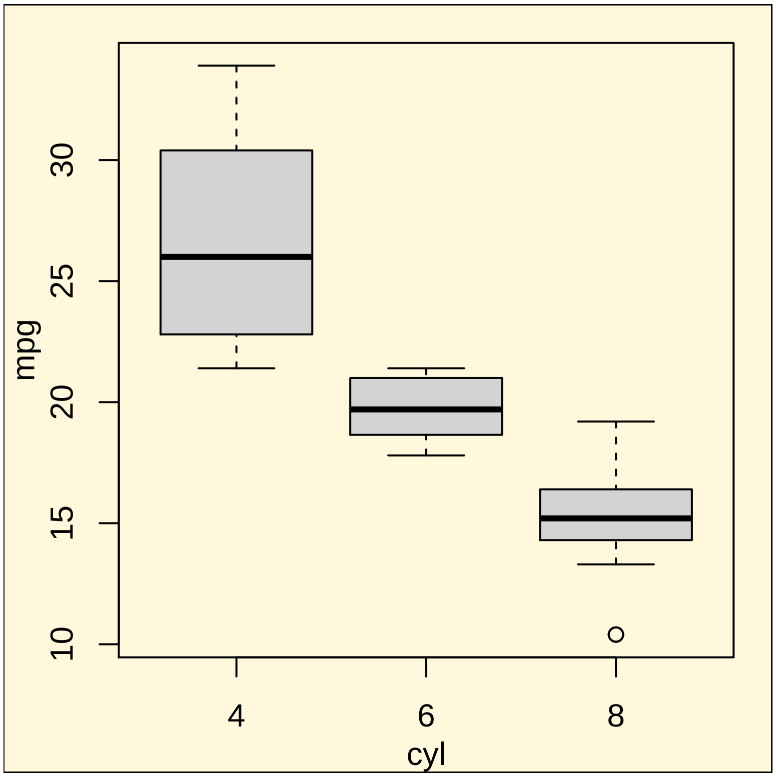
```
library(ggplot2)
library(cowplot)
require(lattice)
p1<- ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.5) +
  scale_y_continuous(expand = expansion(mult = c(0, 0.05))) +
```

```
  theme_minimal_hgrid(10)
p2<- densityplot(~Sepal.Length|Species , iris)

plot_grid(p1, p2,  rel_widths = c(1, 1.5)
          , labels = c('A', 'B'))
```



Mixing different plotting frameworks

```
# require(ggplot2)
# require(cowplot)
# require(lattice)

p1 <- function() {
  par(
    mar = c(3, 3, 1, 1),
    mgp = c(2, 1, 0)
  )
  boxplot(mpg ~ cyl, xlab = "cyl", ylab = "mpg", data = mtcars)
}

ggdraw(p1) +
  theme(plot.background = element_rect(fill = "cornsilk"))
```

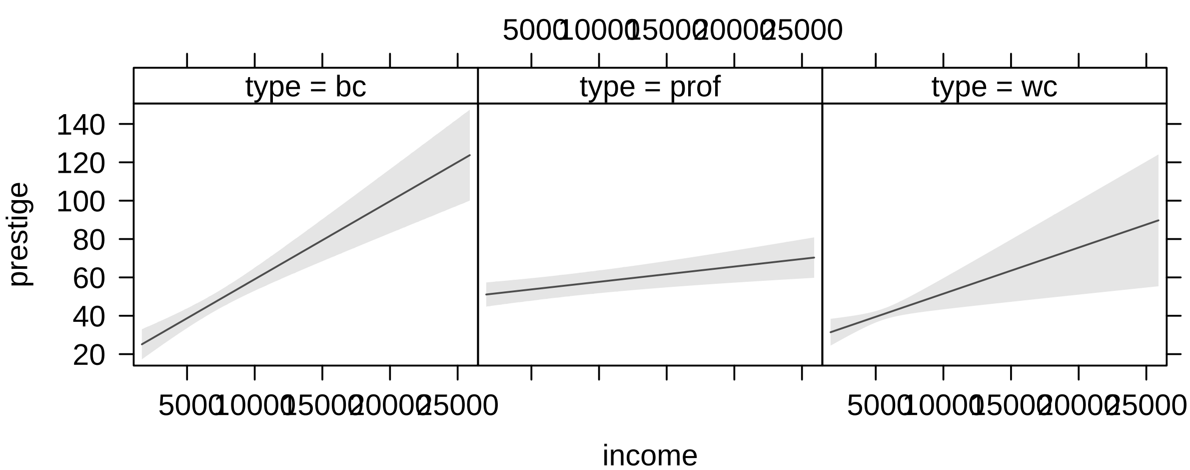


Effectplot mit effect

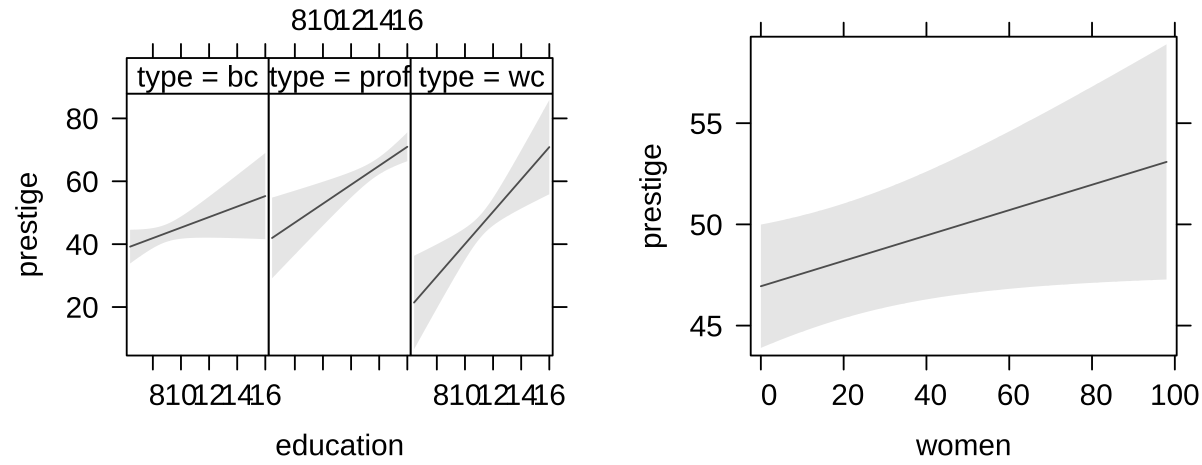
predictorEffect()

Von mir lang ignorierte Variante von Effect mit Formeln!

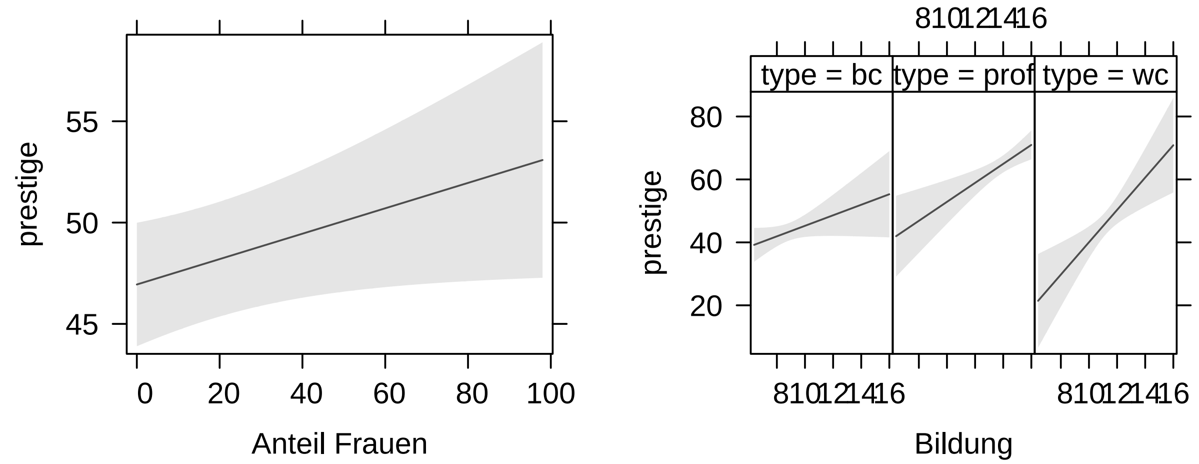
```
mod <- lm(prestige ~ type*(education + income) + women, Prestige)
plot(predictorEffect("income", mod), main="", rug=FALSE)
```



```r
plot(predictorEffects(mod, ~ education + women), main="", rug=FALSE)
```




```r
plot(predictorEffects(mod, ~ women+ education),
     axes= list(x=list( women=list(lab="Anteil Frauen"),
                              education=list(lab="Bildung"))), main="", rug=FALSE)
```

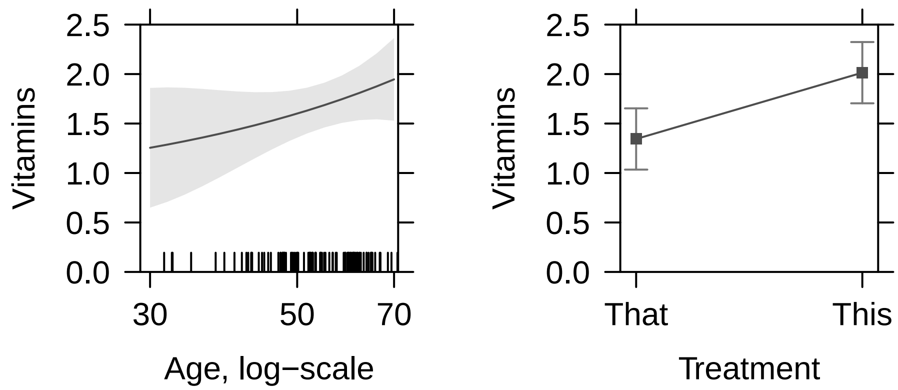


Modifizier plot.efflist

allEffects

```r
ef <- allEffects(lm(A ~ B + C))
plot(ef,
     axes = list(
       x = list(
        B = list(
         transform = list(trans = log, inverse = exp),
         ticks = list(at = c(30, 50, 70)),
         lab = "Age, log-scale"),
       C = list(lab = "Treatment")
```

```
       ),
     y = list(lim= c(.0, 2.5),
              lab = "Vitamins"
              #  transform = list(link = Logit, inverse = invLogit),
              #  transform=list(trans=log, inverse=exp),
              #  type="rescale",
              #  ticks = list(at = c(.05, .25, .50, .75)),
              #
       )),
     main = "")
```



Das ist hingegen obsolet!

```
plot.efflist <- stp25plot:::plot.efflist
ef <- allEffects(lm(A ~ B + C))
plot(ef, xlab = c("Foo", "Bar"), main="Modifiziert")
```

Effectplot mit emmeans

```
library(emmeans)
head(pigs)
```

```
##   source percent conc
## 1   fish       9 27.8
## 2   fish       9 23.7
## 3   fish      12 31.5
## 4   fish      12 28.5
## 5   fish      12 32.8
## 6   fish      15 34.0
```

```
pigs.lm1 <- lm(log(conc) ~ source + factor(percent), data = pigs)
ref_grid(pigs.lm1)
```

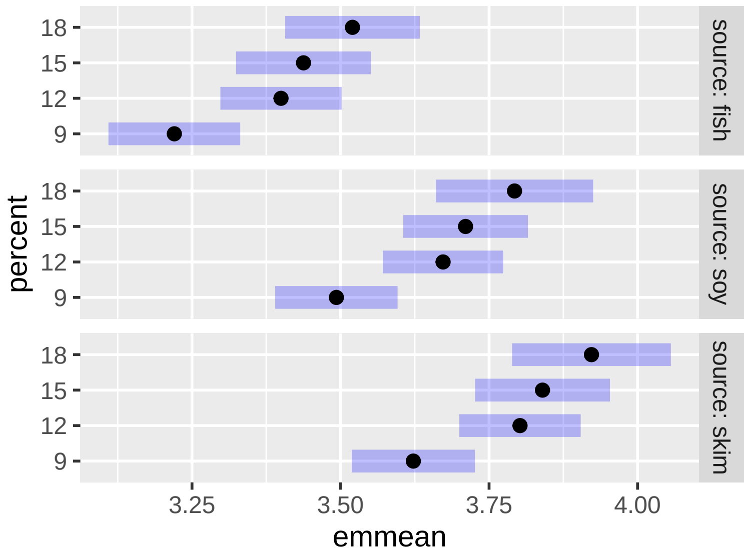
```
## 'emmGrid' object with variables:
##     source = fish, soy, skim
##     percent =  9, 12, 15, 18
## Transformation: "log"
```

```
pigs.lm2 <- lm(log(conc) ~ source + percent, data = pigs)
ref_grid(pigs.lm2)
```

```
## 'emmGrid' object with variables:
##     source = fish, soy, skim
##     percent = 12.931
## Transformation: "log"
```

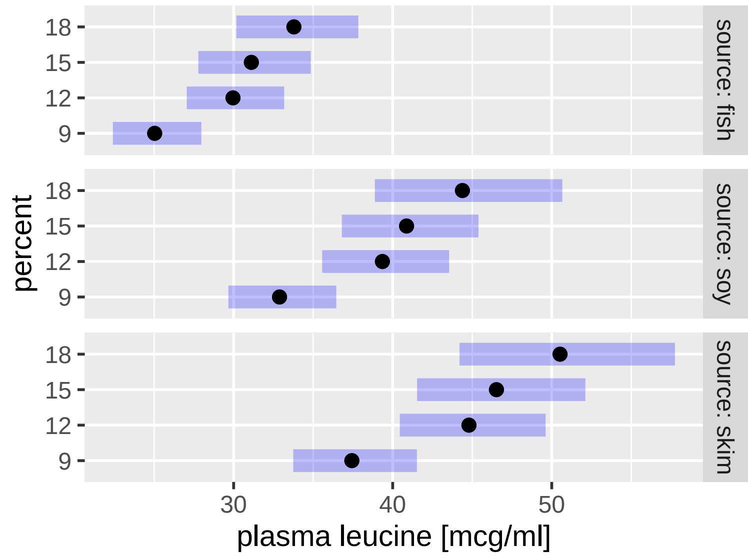
emmeans default

```r
plot(emmeans(pigs.lm1,
             ~ percent | source))
```

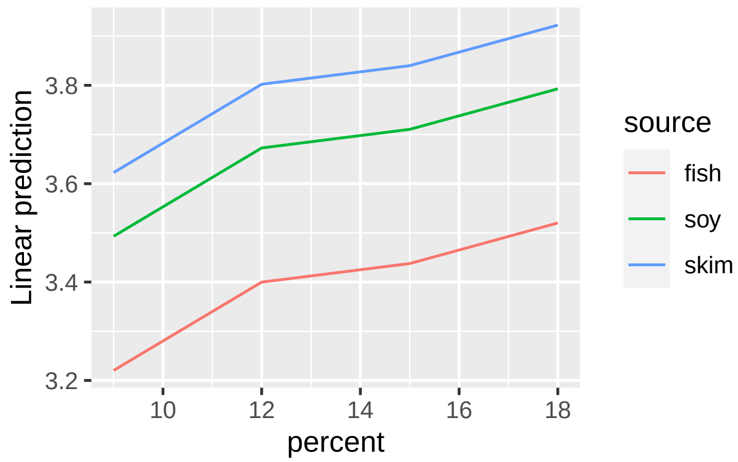


emmeans ruecktransformiert

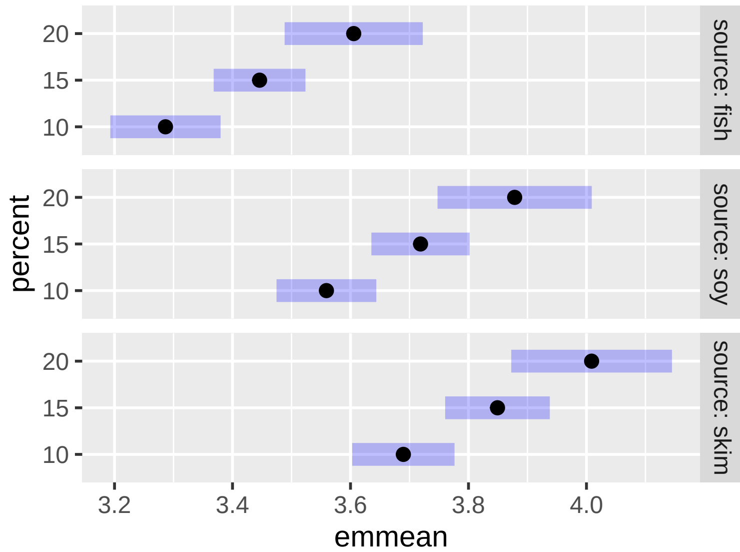
```r
plot(emmeans(pigs.lm1,
             ~ percent | source),
             xlab= "plasma leucine [mcg/ml]" ,
             type = "response")
```



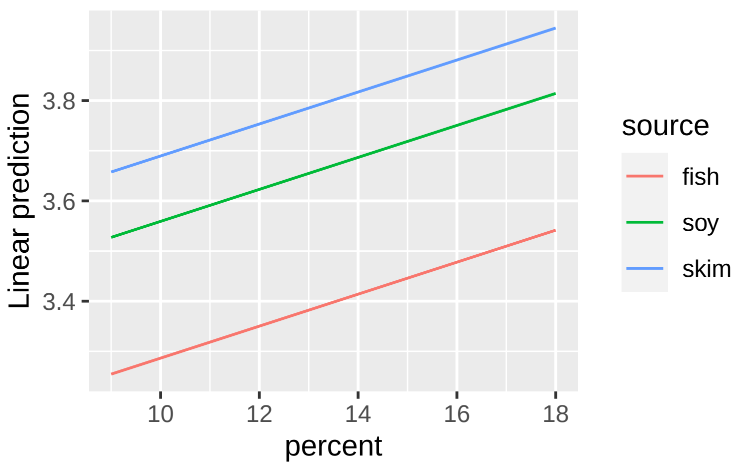

```
emmip(pigs.lm1,
      source ~ percent)
```



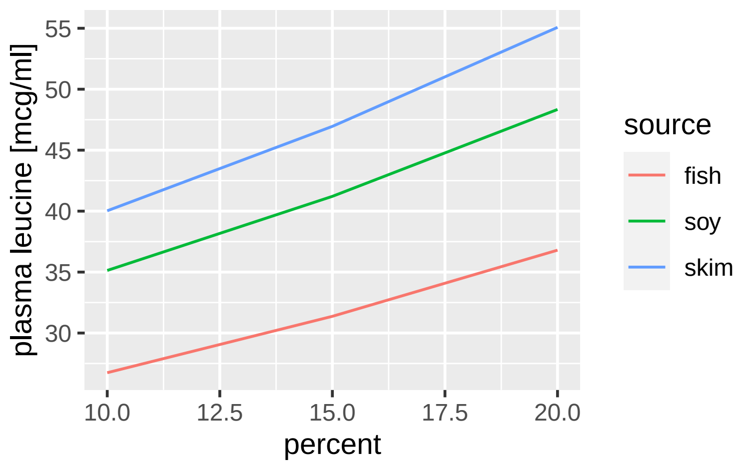

```
plot(emmeans(pigs.lm2,
             ~ percent | source,
             at = list(percent = c(10, 15, 20))
             )
     )
```




```
emmip(
  ref_grid(pigs.lm2, cov.reduce = FALSE),
  source ~ percent)
```

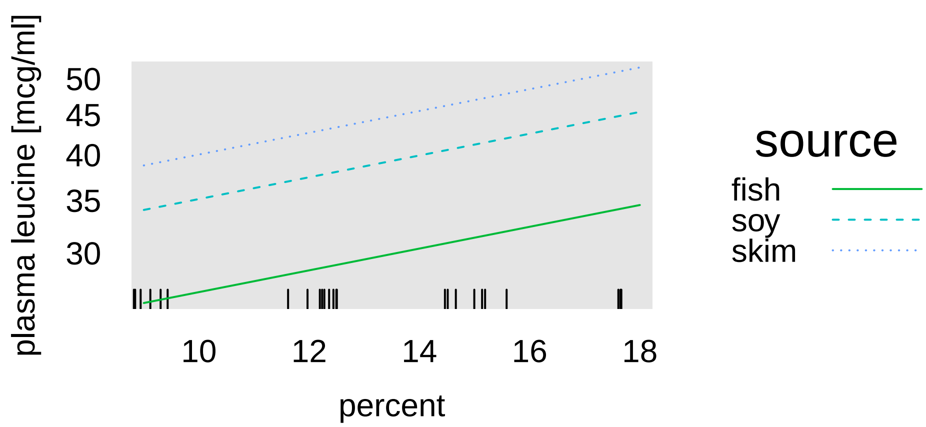



```
emmip(ref_grid(pigs.lm2,
               at= list(percent = c(10, 15, 20))),
      source ~ percent,
      ylab= "plasma leucine [mcg/ml]" ,
      type = "response"
      )
```

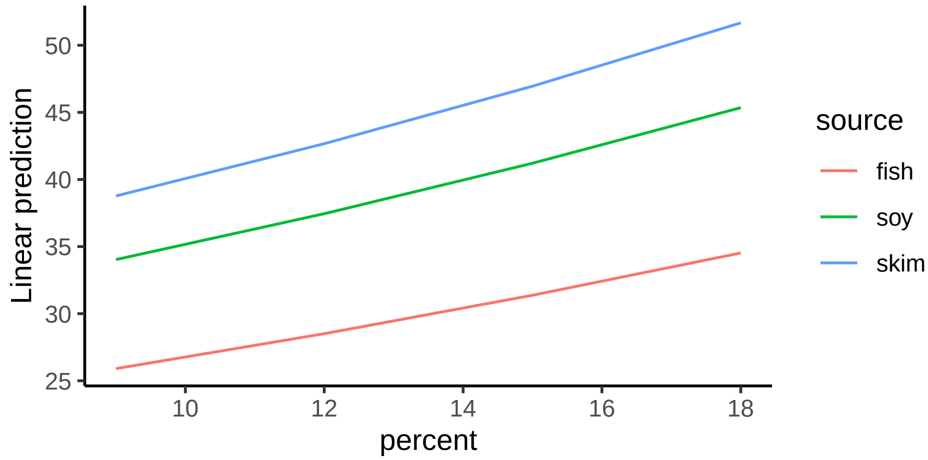


Klassiker mit Effect()

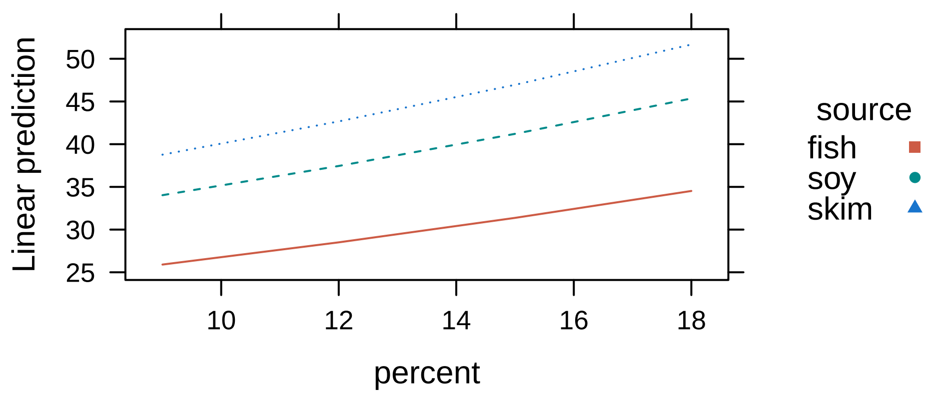
```
set_lattice_ggplot()
plot(Effect(c("source", "percent"),
            pigs.lm2,
            transformation=list(link=log, inverse=exp)),
     multiline=TRUE,
     key.args = list(space="right" ),
     main="",
     ylab="plasma leucine [mcg/ml]")
```



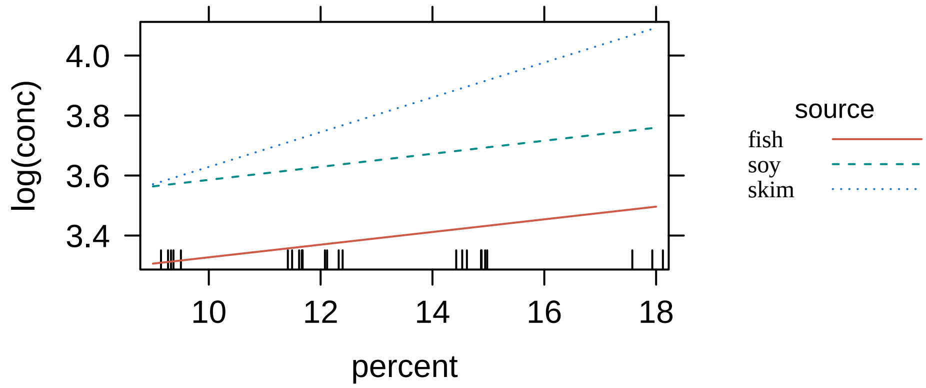

```
emmip(ref_grid(pigs.lm2,
               cov.reduce = FALSE,
               transform = "response"),
      source ~ percent  #,  CIs=TRUE
      ) + ggplot2::theme_classic()
```




```
set_lattice_bw(col = c("coral3", "cyan4", "dodgerblue3"))
emmip(
  ref_grid(pigs.lm2,
           cov.reduce = FALSE,
           transform = "response"),
  source ~ percent,
  engine = "lattice"
)
```



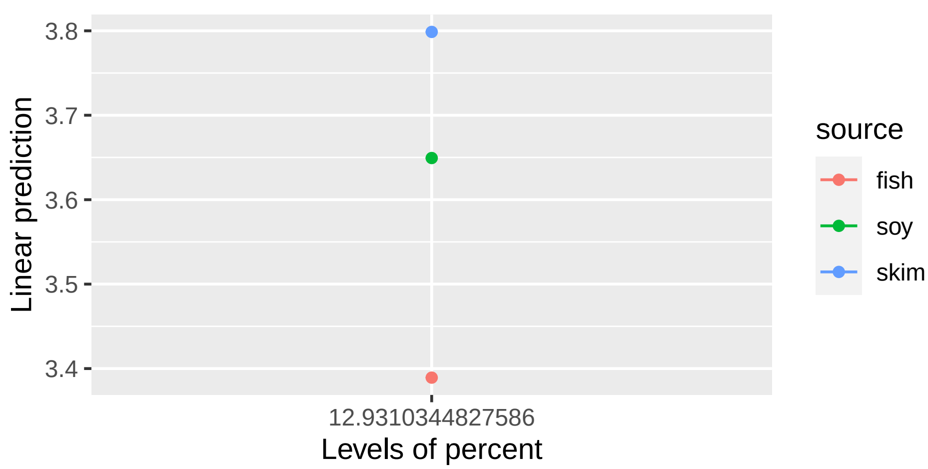

```
pigs.lm3 <- lm(log(conc) ~ source * percent, data = pigs)

plot(
  allEffects(pigs.lm3),
  main = "",
  multiline = TRUE,
  key.args = list(
    space = "right", columns = 1,
    border = FALSE,
    fontfamily = "serif",
    cex.title = .80,  cex = 0.75
  )
)
```



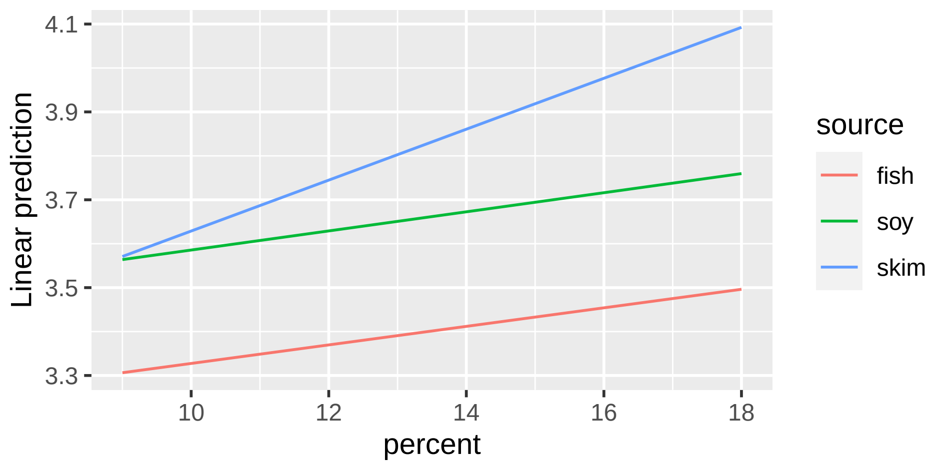

```
emmip(
  ref_grid(pigs.lm3, cov.reduce = TRUE),
  source ~ percent)
```

```
## Suggestion: Add 'at = list(percent = ...)' to call to see > 1 value per group.
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```




```
emmip(
  ref_grid(pigs.lm3, cov.reduce = FALSE),
  source ~ percent)
```

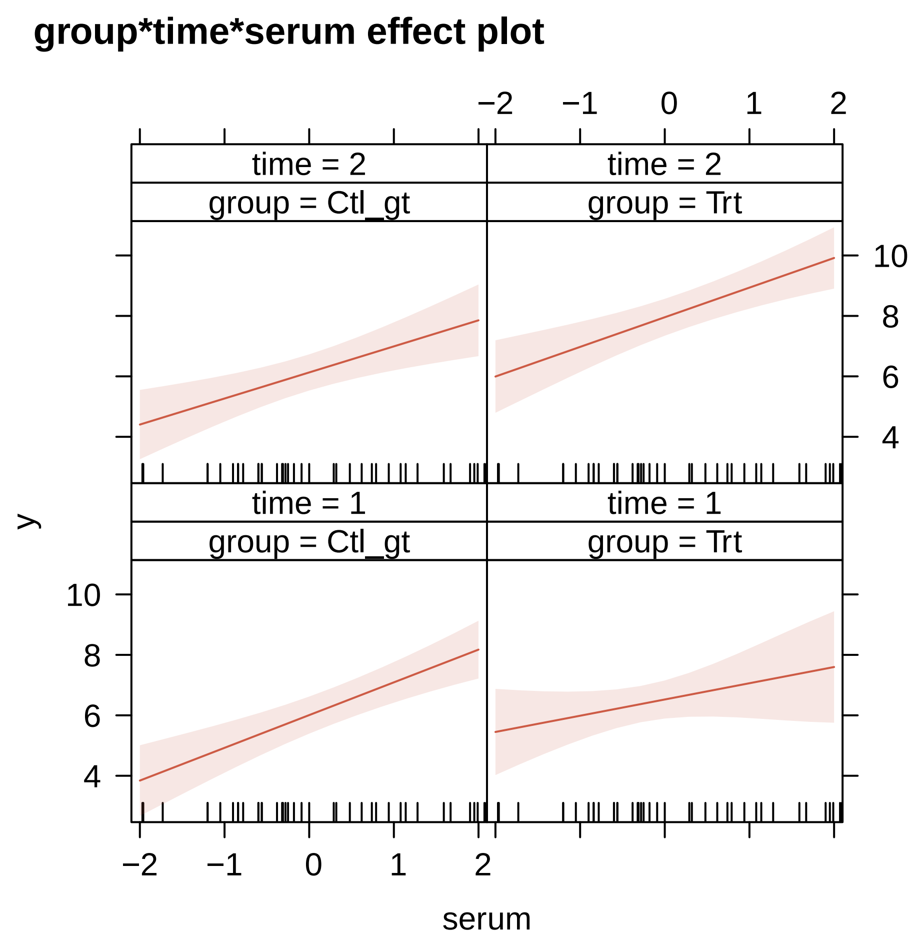


transformation

library(effects) John Fox URL http://www.jstatsoft.org/v32/i01/

```
fit <- lm(y ~ group * time * serum, DF)
```

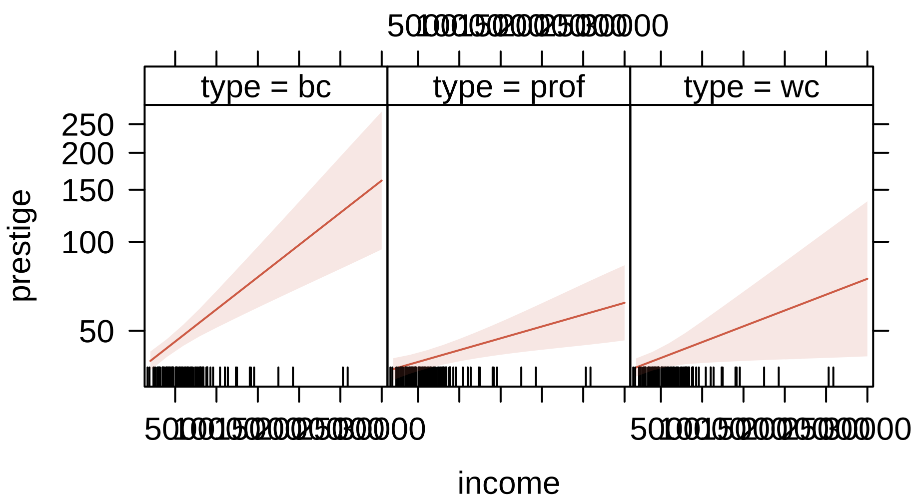
```
plot(effects::allEffects(fit))
```



```r
APA2( ~ log(prestige) + income + type + education,
      data = Prestige,
      output = "text")
```

```
##
##  Tab 1: Charakteristik
##                         Item   n                   m
## prestige   prestige (mean) 102        3.77 (0.39)
## income       income (mean) 102 6797.90 (4245.92)
## type                   type   98
## 1                        bc               45% (44)
## 2                      prof               32% (31)
## 3                        wc               23% (23)
## education education (mean) 102      10.74 (2.73)
##
##
```

```r
mod <- lm(log(prestige) ~ income:type + education, data = Prestige)

# does not work: effect("income:type", mod, transformation=list(link=log, inverse=exp))

plot(Effect(c("income", "type"), mod,
            transformation=list(link=log, inverse=exp)),
     main="", ylab="prestige")
```




```r
set_lattice_bw()
plot(
  Effect(c("time", "group"), fit,
         partial.residuals = TRUE),
  main = FALSE,
  lty = 0,
  partial.residuals = list(pch = 16,
                           col = gray.colors(nrow(DF))[order(DF$serum)])
)
```

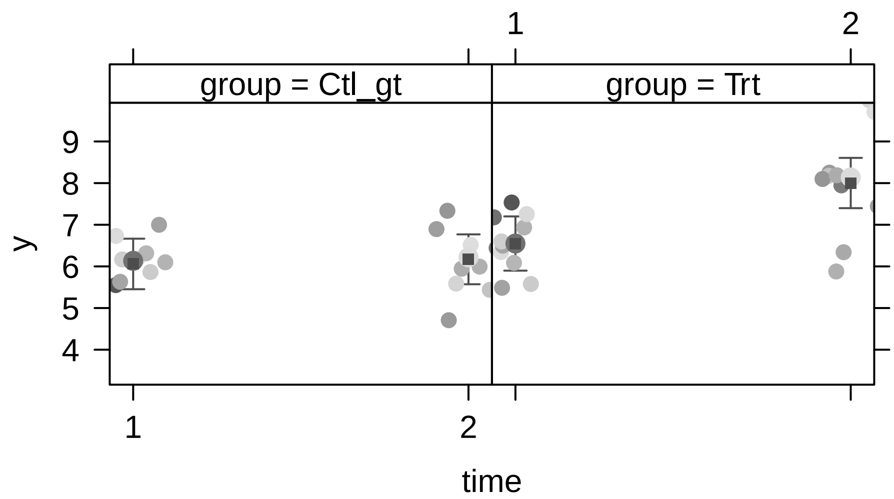


Figure 6: Effect patial.residuals

Effectplot mit ggplot

"'{reffect-ggplot, fig.cap='Effect ggplot',fig.height=3, fig.width= 5}

Model <- lm(drat~hp*cyl, data=mtcars) ef <- effect(term = "hp:cyl", Model, default.levels = 9) # 9 because the breaks are nicer ef2 <- as.data.frame(ef)

ggplot(ef2, aes(hp, fit, col = factor(cyl))) + geom_line() + labs(y = 'drat') + #+ ylim(0, 10) jtools::theme_apa()

```
## GOF-Plots

library(car)


```r
car::residualPlots(fit)
```

```
##              Test stat Pr(>|Test stat|)
## group
## time
## serum          -0.3948           0.6957
## Tukey test     -0.6940           0.4877
```

```r
car::marginalModelPlots(fit)
```

```
## Warning in mmps(...): Interactions and/or factors skipped
```

```r
car::avPlots(fit)
```
```

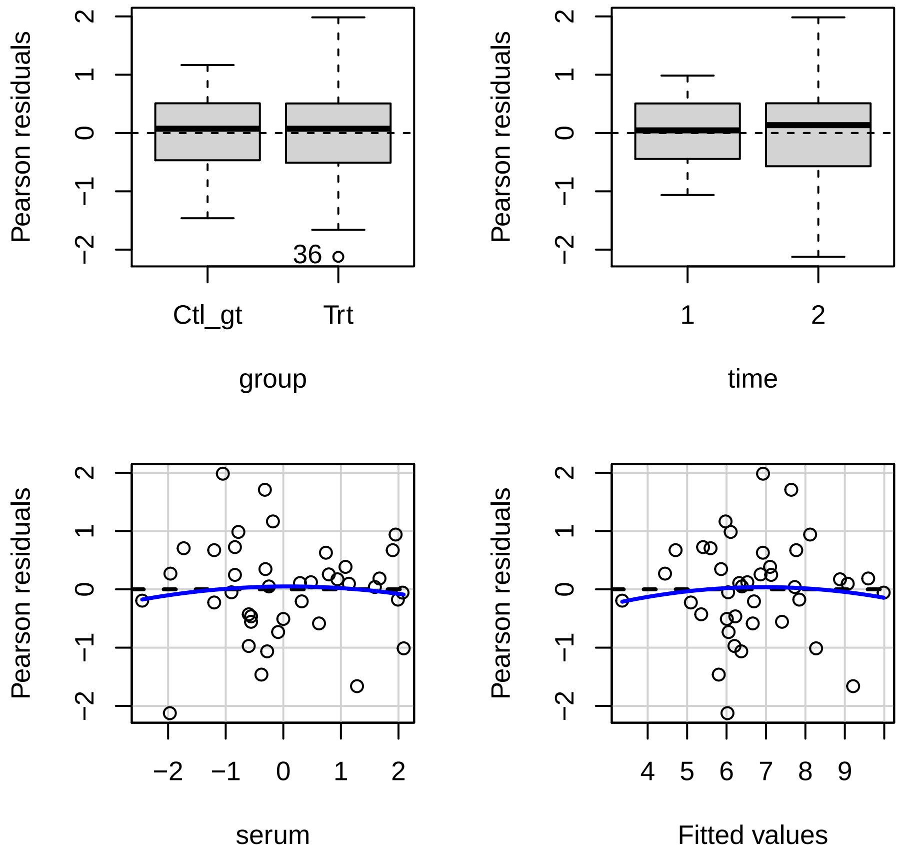


Figure 7: residualPlots

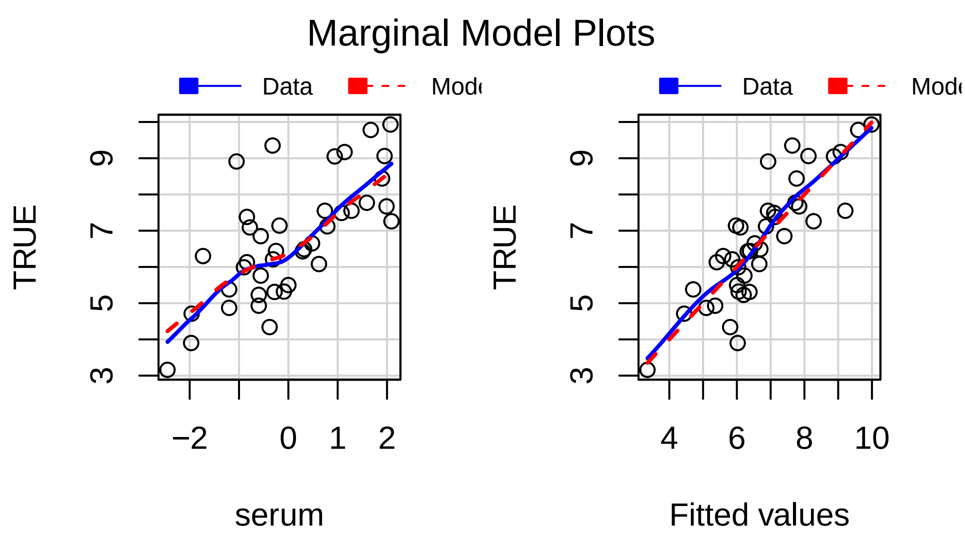


Figure 8: marginalModelPlots

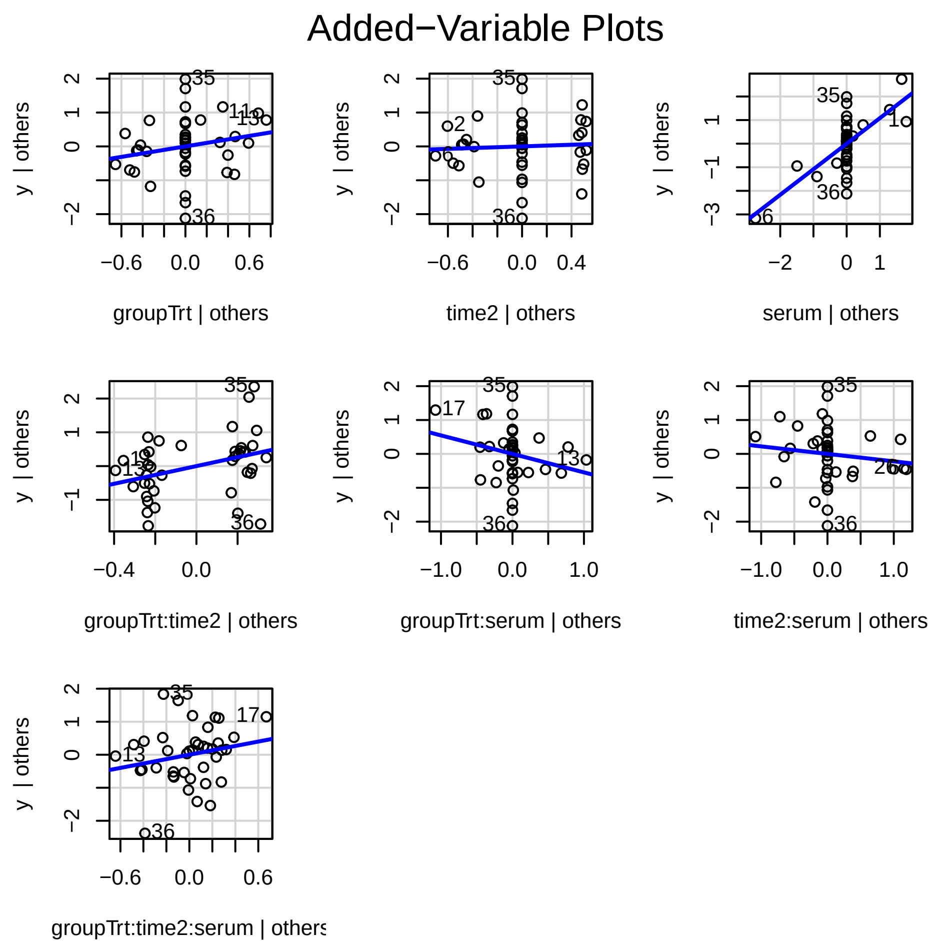


Figure 9: avPlots

library(visreg)

Patrick Breheny and Woodrow Burchett URL: https://cran.r-project.org/web/packages/visreg/vignettes/quick-start.html

```
par(mfrow=c(1,3))
visreg::visreg(fit)
```

```
## Conditions used in construction of plot
## time: 1
## serum: -0.215
```

```
## Conditions used in construction of plot
## group: Ctl_gt
## serum: -0.215
```

```
## Conditions used in construction of plot
## group: Ctl_gt
## time: 1
```

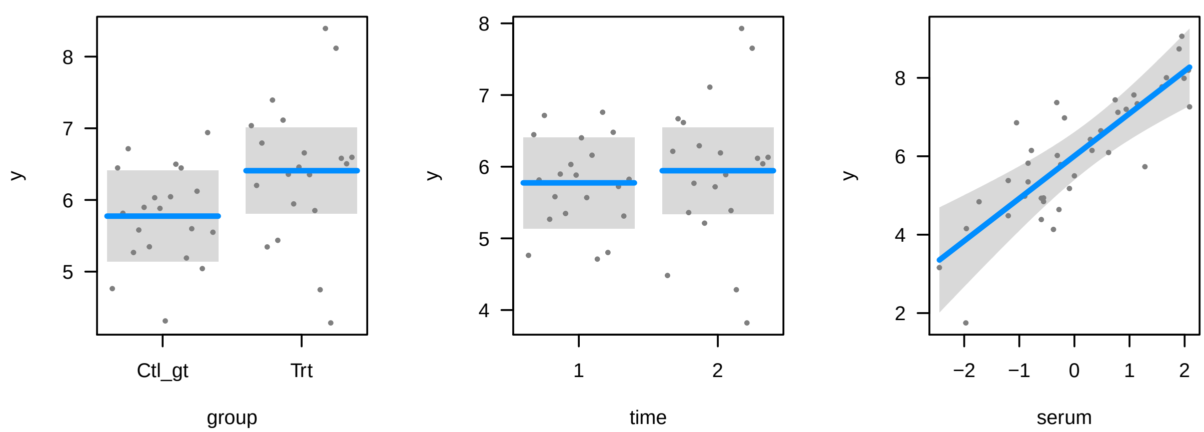


Figure 10: visreg

library(stats) termplot

```
par(mfrow=c(1,3))
stats::termplot(fit,
                se = TRUE,
                resid = TRUE,
                plot=TRUE, ask=FALSE)
```

library(rockchalk) Paul E. Johnson URL https://github.com/pauljohn32/rockchalk

Hier gibt es keine Updates mehr???

```
rockchalk::plotSlopes(fit,
                      plotx = "group",
                      interval = "confidence")
```

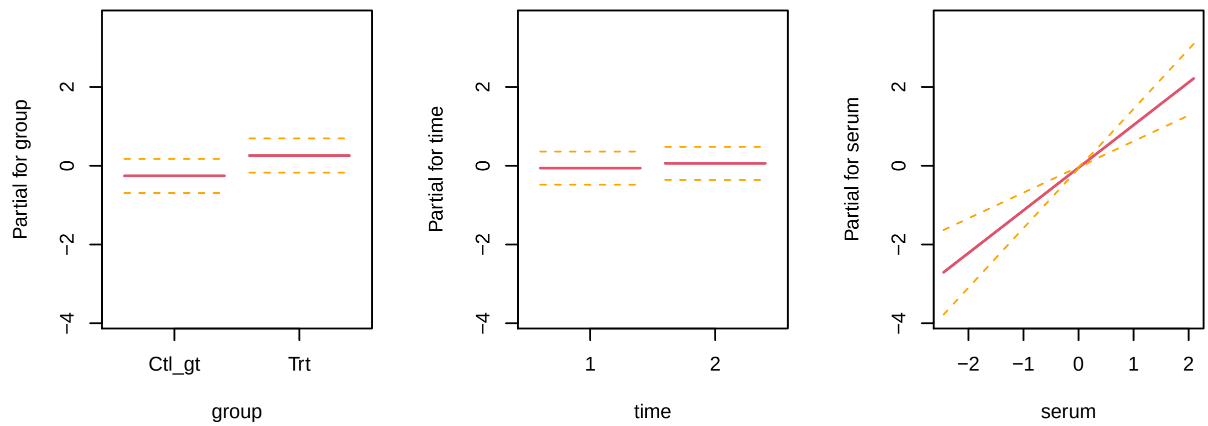


Figure 11: termplot

```
rockchalk::plotSlopes(fit,
                      plotx = "group",
                      modx = "time",
                      interval = "confidence")
```

```
raw_data <-
  data.frame(
    subject_id = rep(1:6, 4),
    time = as.factor(rep(c("t0", "t1"), each = 12)),
    group = rep(rep(c("Control", "Treat"), each = 6), 2),
    value = c(2:7, 6:11, 3:8, 7:12)
  )
head(raw_data)
```

```
##   subject_id time   group value
## 1          1   t0 Control     2
## 2          2   t0 Control     3
## 3          3   t0 Control     4
## 4          4   t0 Control     5
## 5          5   t0 Control     6
## 6          6   t0 Control     7
```

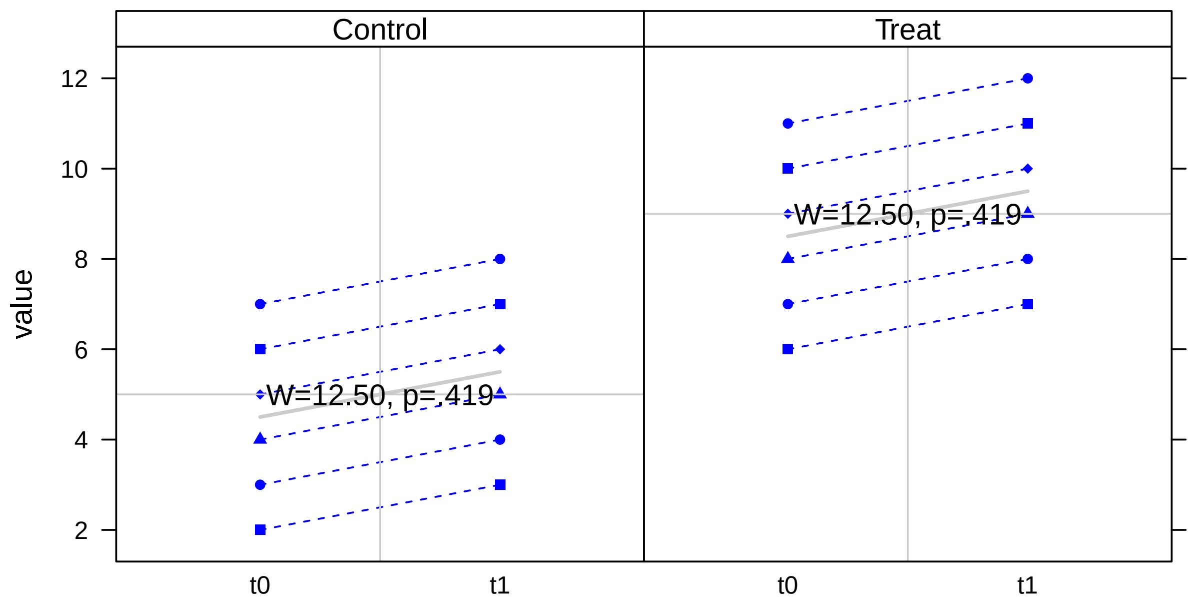
```
stripplot(
  value ~ time | group,
  groups = subject_id,
  data = raw_data,
  panel = function(x, y, ...) {
    panel.stripplot(x,
                    y,
                    type =  "b",
                    col = "blue",
                    lty = 2,
                    ...)
    panel.average(x,
```

```
                  y,
                  fun = mean,
                  lwd = 2,
                  col = "gray80",
                  ...)     # plot line connecting means
    mm <- mean(y)
    panel.abline(h = mm, v = 1.5, col = "gray80")
    panel.text(x = 1.5, y = mm, APA(wilcox.test(y ~ x)))

  }
)
```

```
## Warning in wilcox.test.default(x = 2:7, y = 3:8): cannot compute exact p-value
## with ties

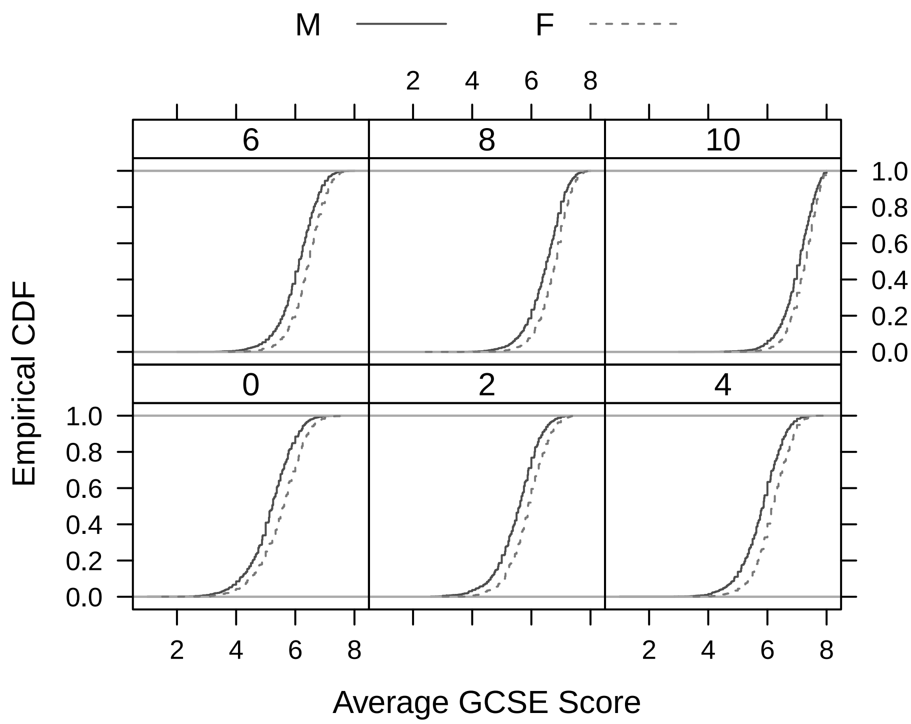
## Warning in wilcox.test.default(x = 6:11, y = 7:12): cannot compute exact p-value
## with ties
```



ECDF-Plot

```
data(Chem97, package = "mlmRev")

ecdfplot(~gcsescore | factor(score), data = Chem97,
    groups = gender,
    auto.key = list(columns = 2),
    subset = gcsescore > 0,
    xlab = "Average GCSE Score")
```



data(singer, package = "lattice")

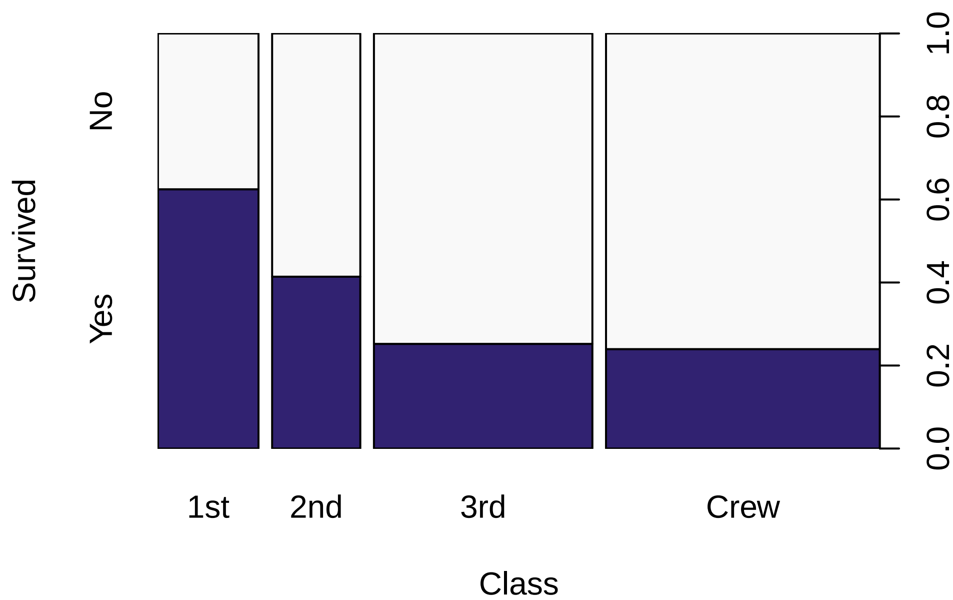
Interessante Grafik Beispiele

Spine Plots and Spinograms

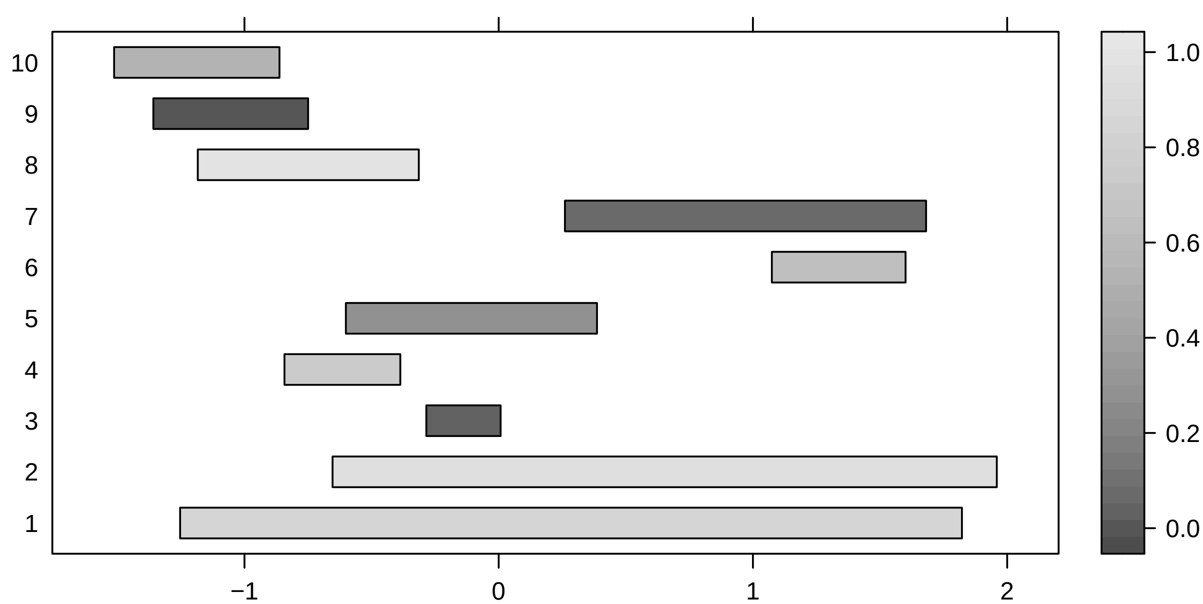
```
library("colorspace")


ttnc <- margin.table(Titanic, c(1, 4))

spineplot(ttnc, col = sequential_hcl(2, palette = "Purples 3"))
```



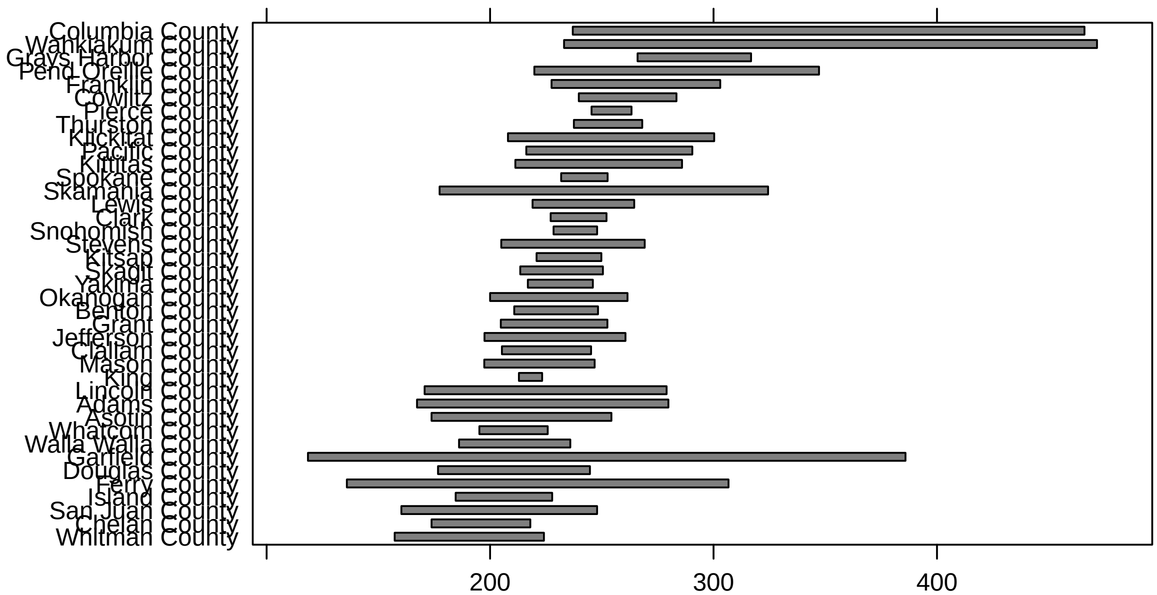

```
require(latticeExtra)
segplot(factor(1:10) ~ rnorm(10) + rnorm(10), level = runif(10))
```



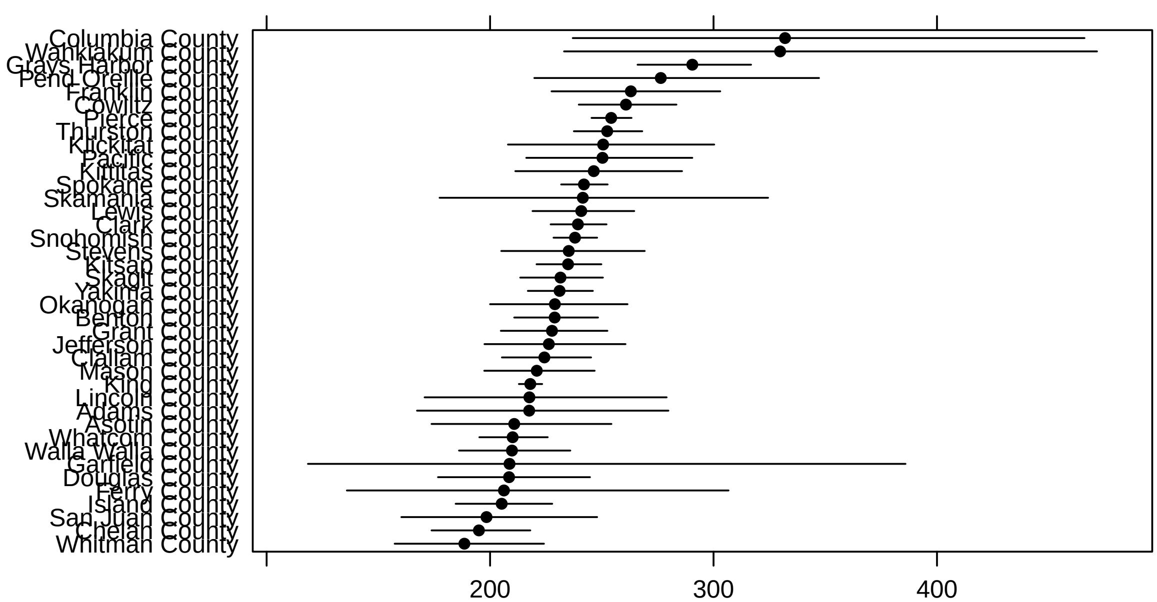

```
data(USCancerRates)

segplot(reorder(factor(county), rate.male) ~ LCL95.male + UCL95.male,
```

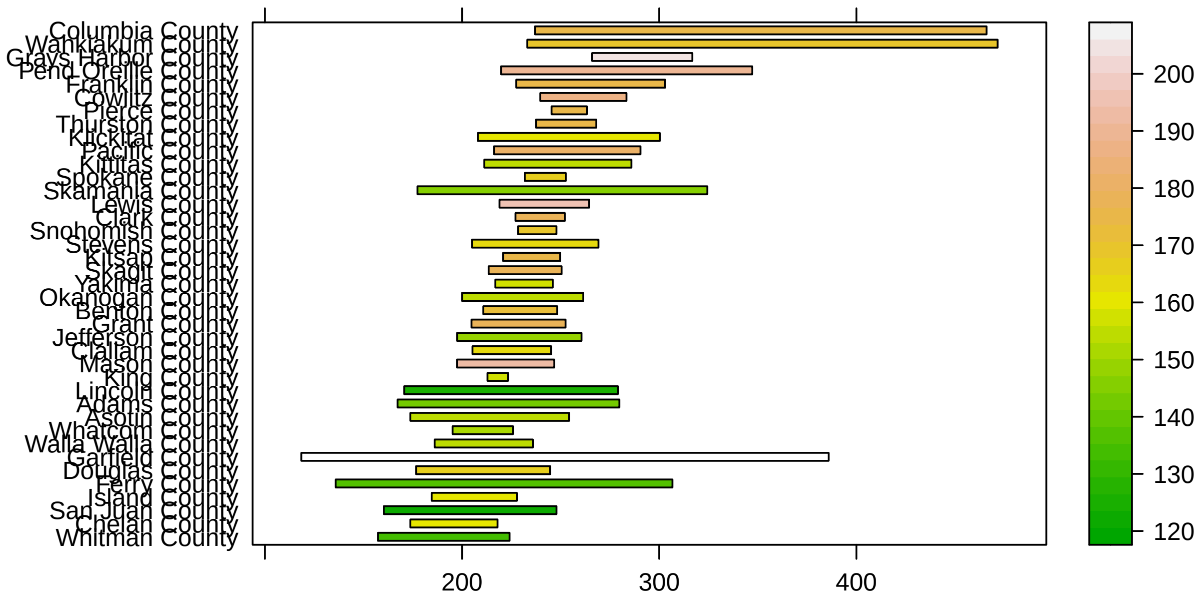
```r
        data = subset(USCancerRates, state == "Washington"))
```



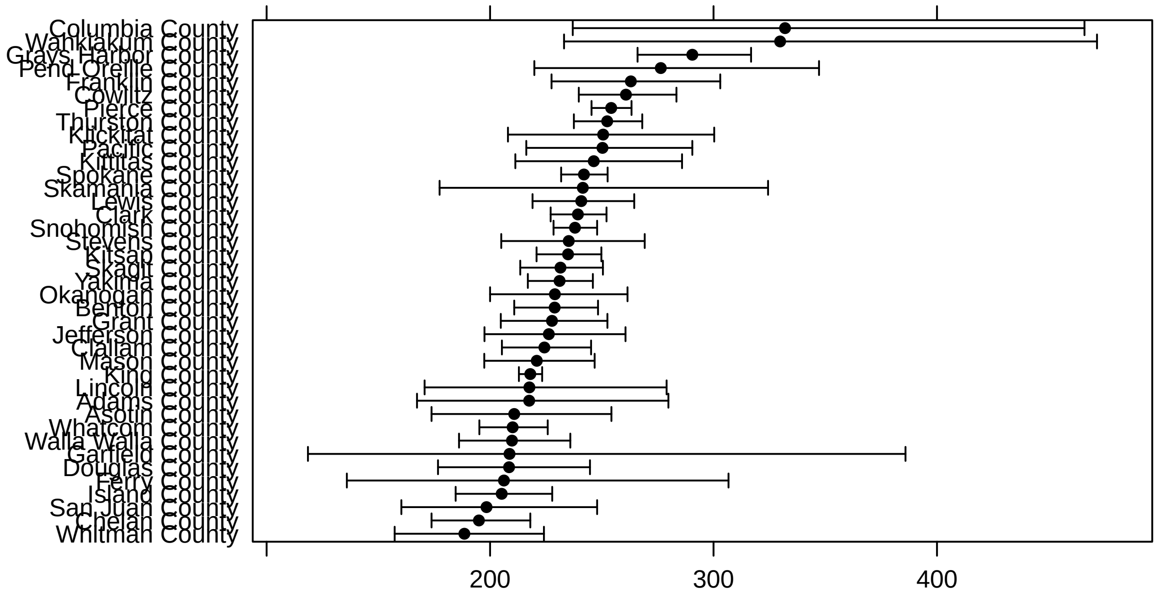

```r
segplot(reorder(factor(county), rate.male) ~ LCL95.male + UCL95.male,
        data = subset(USCancerRates, state == "Washington"),
        draw.bands = FALSE,
        centers = rate.male)
```



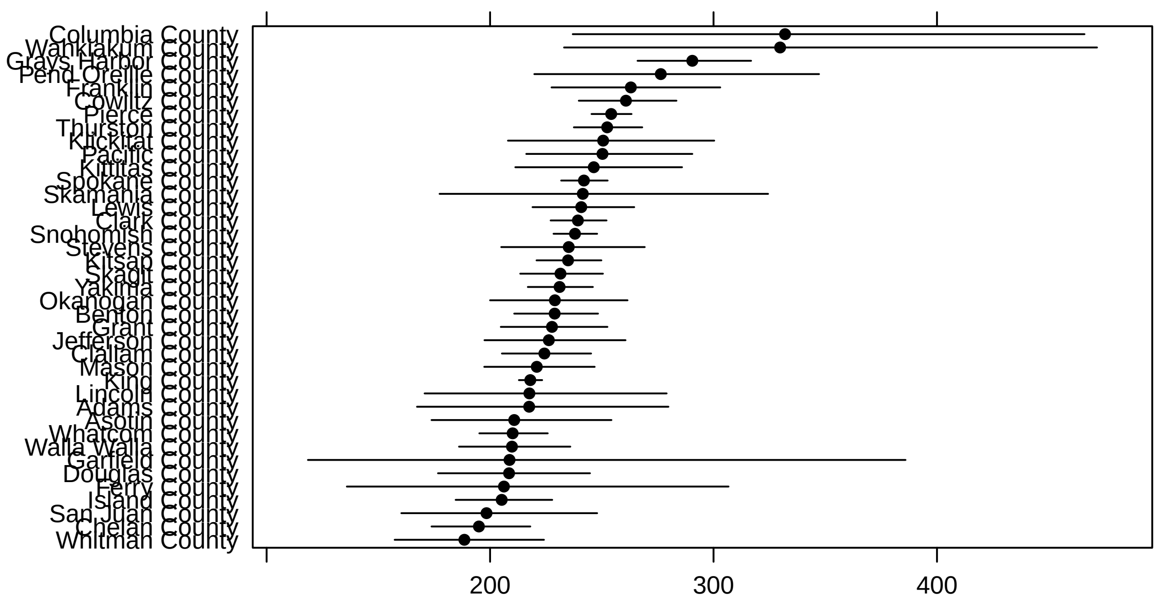
```r
segplot(reorder(factor(county), rate.male) ~ LCL95.male + UCL95.male,
        data = subset(USCancerRates, state == "Washington"),
        level = rate.female,
        col.regions = terrain.colors)
```




```r
segplot(reorder(factor(county), rate.male) ~ LCL95.male + UCL95.male,
        data = subset(USCancerRates, state == "Washington"),
        draw.bands = FALSE,
        centers = rate.male,
        segments.fun = panel.arrows,
        ends = "both",
        angle = 90,
        length = 1,
        unit = "mm")
```




```r
segplot(reorder(factor(county), rate.male) ~ LCL95.male + UCL95.male,
        data = subset(USCancerRates, state ==  "Washington"),
        draw.bands = FALSE, centers = rate.male)
```



Links

https://ggobi.github.io/ggally/index.html http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/78-perfect-scatter-plots-with-correlation-and-marginal-histograms/

ggpubr

http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/78-perfect-scatter-plots-with-correlation-and-marginal-histograms/