. ( ) !

. , , ,
.

. , .
. .

(,)

(,) . y y , x
. ( ) (:) .

: . , .
y , y .

```r
# ---
# Data set creation.

set.seed(93384)

time <- c(0, 0.5, 1, 2, 4, 8, 12, 16, 24)
n <- 32 # no of subjects

data <- expand.grid(ID = 1:n, time = time)

bw <- data.frame(
  ID = sort(unique(data$ID)),
  bw = rlnorm(n, log(75), sdlog = 0.25)
)

bw$bw.category <- cut(bw$bw,
  breaks = quantile(bw$bw, c(0, 0.33, 0.66, 1)),
  labels = paste(c("low", "medium", "high"), "body weight"),
  include.lowest = TRUE
)

data <- merge(data, bw)

data <- data[order(data$ID, data$time), ]

# Simulate drug concentrations as a function of body weight.
data$conc <- 100 / (data$bw^1.0) * exp(-0.085 * data$time) *
  rlnorm(nrow(data), sdlog = 0.25) + # res. error
  (data$ID - mean(data$ID)) / mean(data$ID) / 4 # r. eff

# ---
# Visualisation.
library(ggplot2)

gg <- list()

data$ID <- factor(data$ID)

gg[["3x1"]] <- ggplot(data, aes(x = time, y = conc, group = ID, color = ID)) +
  geom_line()
gg[["3x1"]] <- gg[["3x1"]] + scale_x_continuous(breaks = seq(0, 24, by = 4))
gg[["3x1"]] <- gg[["3x1"]] + theme_bw() +
  xlab("time [h]") +
  ylab("drug concentration [ng/mL]")
gg[["3x1"]] <- gg[["3x1"]] + facet_grid(bw.category ~ .)
gg[["3x1"]] <- gg[["3x1"]] + theme(legend.position = "none")

gg[["1x3"]] <- gg[["3x1"]] + facet_grid(. ~ bw.category)

# Add space to the rhs of the first figure for better separation in the cowplot.
gg[["3x1"]] <- gg[["3x1"]] +
  theme(plot.margin = unit(c(0.5, 4, 0.5, 0.5), "lines"))

# Both figures into a single output figure.
library(cowplot)
```

```r
plot_grid(gg[[1]], gg[[2]], rel_widths = c(1.5, 2))
```

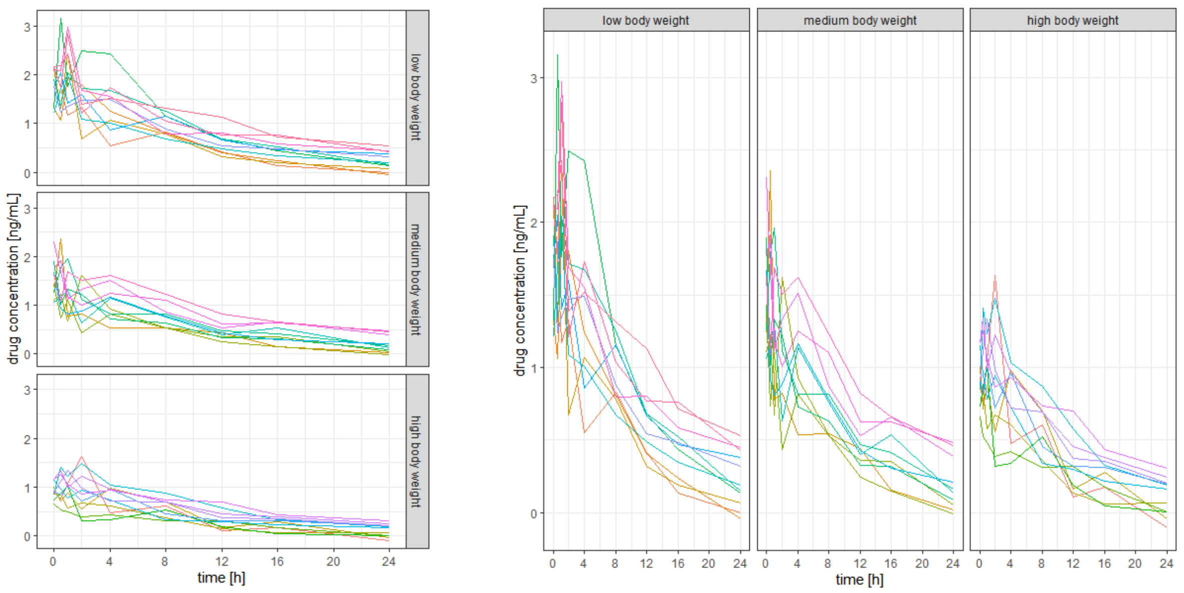

Figure 1: . 3 . 3 .

(Aspect ratio)

x y . . , , (kg m) . 1:1 (x y 1) . x y 1:1 .

: , ( ) . x , y , 1:1 .

```r
# Observed vs predicted (any data with comparable x and y will do).

# ---
# Data set.
# Old Faithful Geyser (Yellowstone) data set with eruption duration
#   and waiting time to the next eruption (both in minutes).
data <- data.frame(
  x = faithful$eruptions,
  y = faithful$waiting
)
```

```
# ---
# Regression model fit.
fit <- lm(y ~ x, data = data)

# Addition of predicted values to the data set.
data$pred <- predict(fit)

# Range of y and y predicted combined.
r <- range(unlist(data[c("y", "pred")]))

# ---
# Plotting.

library(ggplot2)

gg <- ggplot(data, aes(x = pred, y = y))

# Adding the line of identity, y = x
# (note: plotting it first will add points on top).
gg <- gg + geom_abline(intercept = 0, slope = 1, color = "black", linewidth = 1)

# Adding points, removing grey background.
gg <- gg + geom_point() + theme_bw()

# Adding regression fit (local smoother, loess) of y~x.
gg <- gg + geom_smooth(method = "loess", color = "firebrick", se = FALSE)

# Adding axis labels.
gg <- gg + xlab("predicted") + ylab("observed")

# Aspect ratios are commonly not fixed but adapted to figure size.
#   With dynamic displays, the point of different perception
#   might not be obvious depending on the figure/screen size.
#   To make that point independent of figure height and width,
#     the aspect ratio is fixed in this example.
gg <- gg + coord_fixed(ratio=0.5)

# Copy the figure and fix the aspect ratio to 2, i.e.,
#   one pixel in x corresponds to 2 pixels in y.
gg2 <- gg + coord_fixed(ratio=2)

# Setting the aspect ratio to 1 (1 unit in x and y
#   corresponds to the same number of pixels) and
#   setting axis limits to be identical.
gg3 <- gg + coord_fixed(ratio=1, xlim=r, ylim=r)

# Cow (column-wise) plot, combine all figures into one.
library(cowplot)
plot_grid(gg, gg2, gg3, rel_widths = c(4, 2, 3), nrow = 1)
```

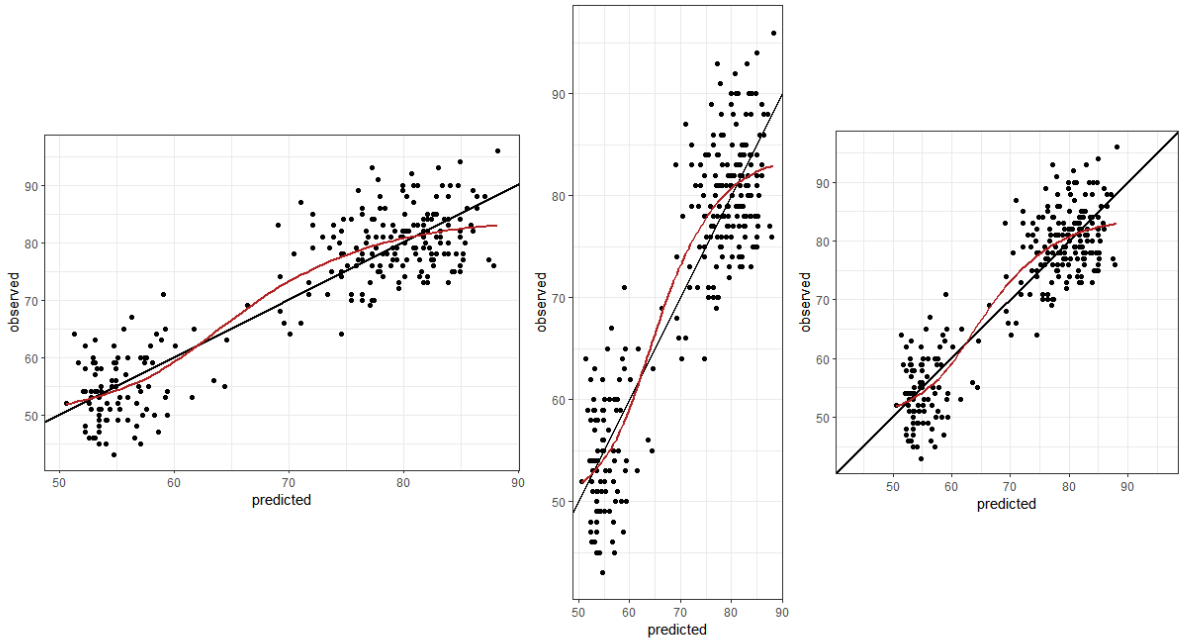

Figure 2: .

(Lines)

, . . , (:
,).

(Points)

. , .
(jittering) .

(Colours)

. . Tufte “
” “ . ” (Tufte 2001, 154).

(Axes)

0 . .
(: 1/4 1 4), . .
. (: 0.25 “1/4”).

x y x y .

: (y) , . y .

```
library(ggplot2)
plot_data <- data.frame(
  type = factor(
    c("Our product", "Competitor"),
    levels = c("Our product", "Competitor")
  ),
  value = c(220, 210)
)

# Original plot
ggplot(plot_data) +
  geom_col(
    mapping = aes(x = type, y = value),
    fill = "lightblue",
    colour = "black"
  ) +
  scale_y_continuous(breaks = seq(0, 220, by = 20), expand = c(0, 0)) +
  labs(x = "", y = "") +
  theme_minimal()

# Offset the y axis
offset <- 208
ggplot(plot_data) +
  geom_col(
    mapping = aes(x = type, y = value - offset),
    fill = "lightblue",
    colour = "black"
  ) +
  scale_y_continuous(
    breaks = seq(0, 14, by = 2),
    labels = seq(0 + offset, 14 + offset, by = 2),
    expand = c(0, 0)
  ) +
  labs(x = "", y = "") +
  theme_minimal()
```

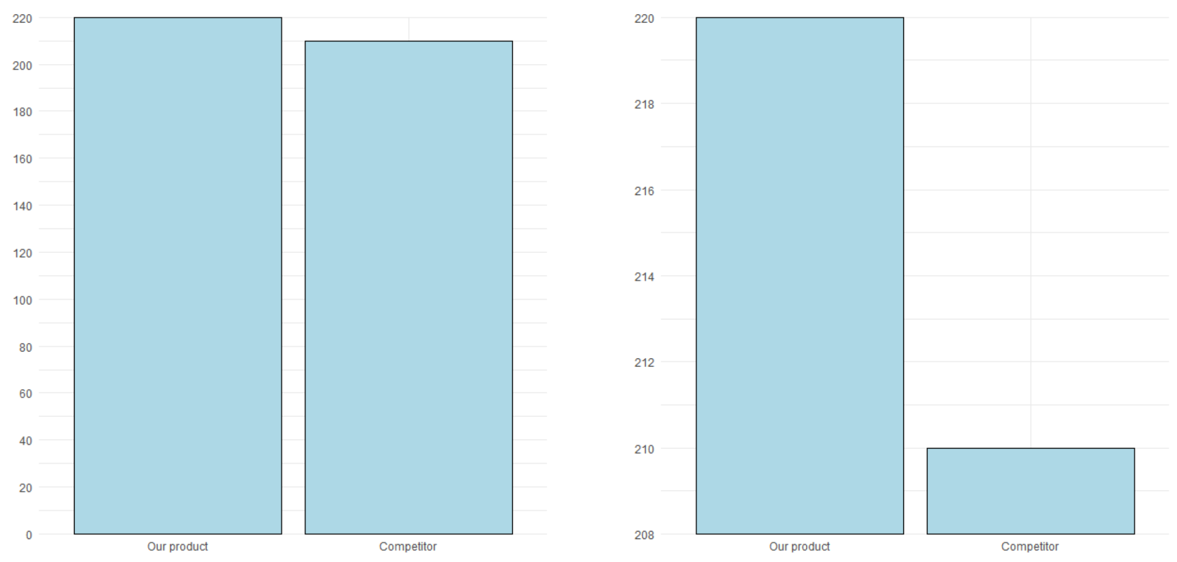

Figure 3: .

: y x ( ) . , ?
( ) (,) . .
( ) . .
.

```
# Naïve plot of y vs x. If there is no change (on average),
# half the data are below the line of no change.
# Asymmetric view, and it depends on y/x or x/y.

set.seed(33838)
```

```r
x <- data.frame(
  x = rlnorm(200, 2, 0.2),
  y = rlnorm(200, 0.2, 0.75)
)
# Add an outlier manually.
x <- rbind(x, data.frame(x = quantile(x$x, 0.8), y = max(x$y) * 1.5))

# ---
# Plotting.

library(ggplot2)
# Scatterplot of y vs x.
gg <- ggplot(x, aes(x = x, y = y)) +
  geom_point() +
  theme_bw()
gg <- gg + geom_hline(yintercept = 1, color = "firebrick", linewidth = 2)
gg <- gg + xlab("x-variable") + ylab("Fold-change")
gg
# Logarithmic axes, symmetric range (!):
xbr <- c(1 / 10, 1 / 5, 1 / 2, 1, 2, 5, 10)
gg <- gg + scale_y_continuous(
  breaks = xbr, trans = "log10",
  limits = max(abs(x$y))^c(-1, 1)
)
gg

# Second axis:
gg <- gg + scale_y_continuous(
  breaks = xbr,
  labels = paste(100 * xbr, "%", sep = ""),
  trans = "log10",
  limits = max(abs(x$y))^c(-1, 1),
  sec.axis = sec_axis(
    trans = ~ . * 1, breaks = xbr,
    labels = ifelse(xbr < 1, paste("1/", 1 / xbr, sep = ""), xbr)
  )
)

# ---
# Univariate distribution (histogram).

gg <- ggplot(x, aes(x = y)) +
  theme_bw() +
  xlab("Fold-change")
gg <- gg + geom_histogram(color = "firebrick", fill = "gray")
gg

# Symmetric range, log scale.
gg <- gg + scale_x_continuous(
  breaks = xbr,
  labels = ifelse(xbr < 1, paste("1/", 1 / xbr), xbr),
  trans = "log10",
  limits = max(abs(x$x))^c(-1, 1)
)
gg
```

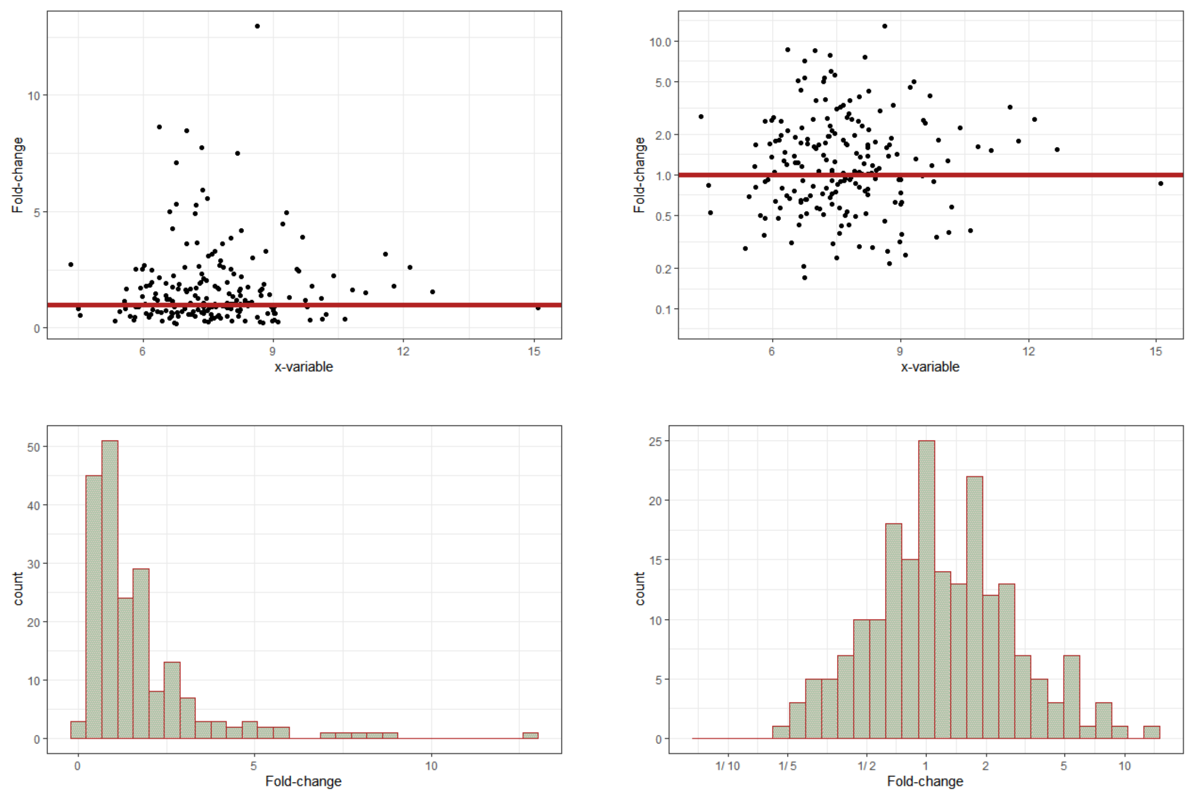

Figure 4: ( ) ( ) .

(Symbols)

(: “+”, “–”, “O”). , .
, , . (: (, , , ,)) .

(Legends)

. .
(:) .
: . .

```r
# ---
# EU stock markets, year and indices DAX, SMI, CAC, and FTSE.

# Store graphics into a list.
gg.list <- list()

# Prepare the data set (reformat EuStockMarkets that comes with R).
x <- EuStockMarkets
df <- data.frame(
  time = as.vector(time(x)),
  index = rep(colnames(x), rep(nrow(x), ncol(x))),
  value = as.vector(x),
  stringsAsFactors = TRUE
)
df$index2 <- df$index # For use with labels later.

library(ggplot2)

# Standard layout and legend.
gg <- ggplot(df, aes(x = time, y = value, group = index, color = index, label = index2))
gg <- gg + geom_line() + theme_bw()

# Nicer axis tick mark settings.
ax <- pretty(df$time, n = 10)
gg <- gg + scale_x_continuous(limits = range(ax), breaks = ax)
gg <- gg + xlab("year") + ylab("Stock index")

gg.list[[1]] <- gg
```

```
# Use the last element of each time series for x,y of the label.
# Use that the last element is the first element of the reversed order,
#   and extract the first element per index by using !duplicated.
y <- df[rev(order(df$time)), ] # descending in time.
y <- y[!duplicated(y$index), ] # first entry per index
y$index2 <- y$index # Create a copy that contains formatted strings.
levels(y$index2)[levels(y$index2) == "FTSE"] <- "\n\nFTSE"
# Add a newline to separate FTSE from DAX.
# Note that the factor level is modified, not the data.

# Drop the legend, move labels into figure.
gg <- gg + geom_text(data = y, hjust = "left", nudge_x = 0.1)
# aes as before, nudge adds space on the lhs.
gg <- gg + theme(legend.position = "none")
gg.list[[2]] <- gg

# ---
# Both figures into a single output figure.

library(cowplot)
plot_grid(gg.list[[1]], gg.list[[2]], rel_widths = c(2.25, 2))
```

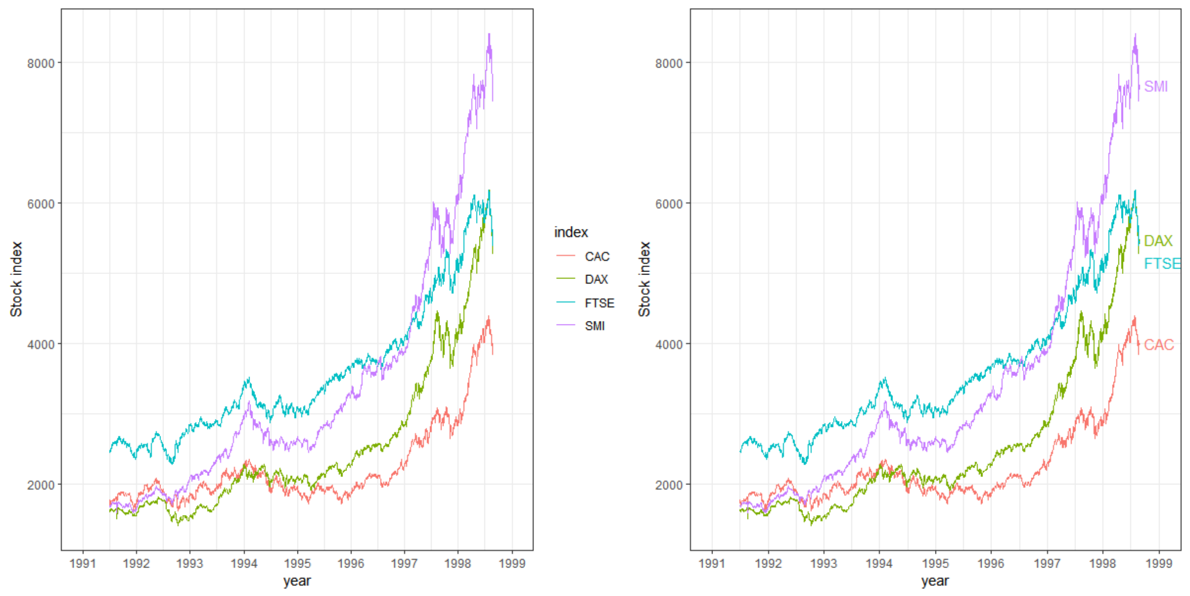

Figure 5: .

(Orientation)

, ( ) (Few 2004, 182). . , , .

. . x .

```
# Store figures into a list.
gg.list <- list()

library(ggplot2)

x <- mpg # miles per gallon data set.
x$car <- paste(x$manufacturer, x$model)

gg <- ggplot(x, aes(x = car, y = hwy, group = car))
gg <- gg + geom_boxplot() + theme_bw() + xlab("Miles per gallon (highway)")
gg.list[["vertical orientation"]] <- gg

gg.list[["horizontal orientation"]] <- gg + coord_flip()

# ---
# Both figures into a single output figure.

library(cowplot)
plot_grid(gg.list[[1]], gg.list[[2]], rel_widths = c(2, 2.5))
```

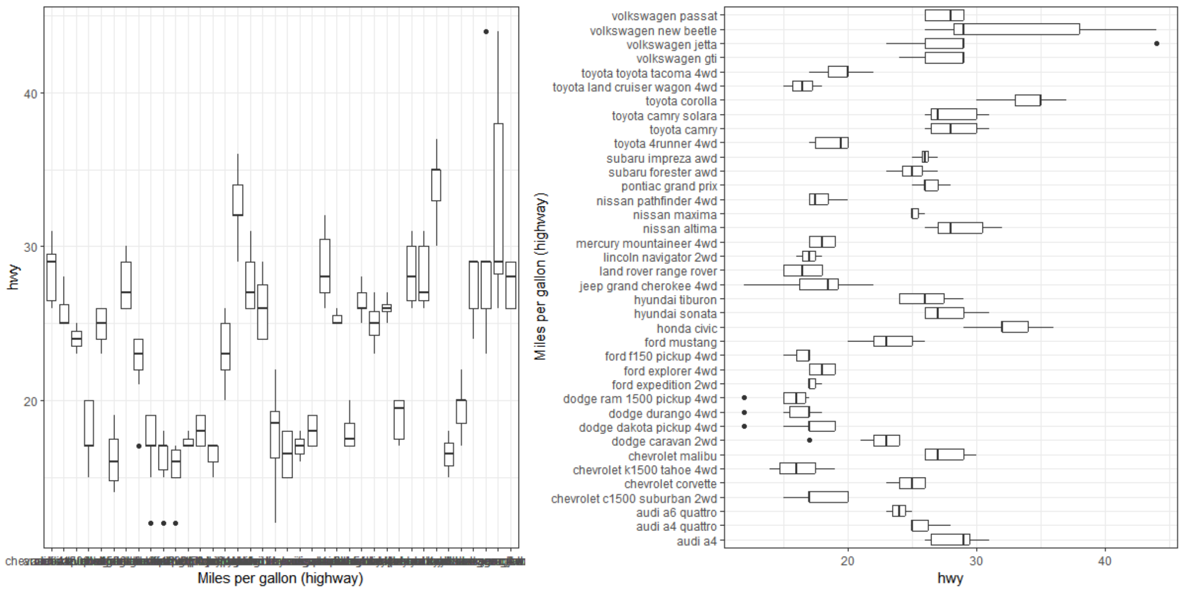

Figure 6: .

(Auxiliary elements)

Tufte “ (chart junk)” , . . , (, ,) . . x y 5 ( ) . y . y .

. x=0(: 0) , y=0(:) (y=x, : x y) . ( ).

(loess, lowess, polynomial) . ( ) .

: (0) . ( ) y (y=0) . ( ) , ( ) ( ) , ( ) .

```
# Function for data set generation.
make.data <- function(
    x = c(0, 0.5, 1, 2, 4, 8, 12, 16, 24),
    y = exp(-0.2 * x) - exp(-0.21 * x),
    sd = 0.25, # std dev of y
    seed = 4384590,
    n = 50) {
  # Setting the random number seed for reproducibility.
  set.seed(seed)
```

```r
  # Creation of x- and y-variables.
  x2 <- rep(x, n)
  y2 <- NULL
  for (i in 1:n) {
    y2 <- c(y2, y * (2 * (n / 4 - i)) + rlnorm(length(y), sd = sd))
  }

  # Creation of an identifier for each profile.
  ID <- factor(rep(1:n, rep(length(y), n)))

  # Composition of the data set.
  df <- data.frame(PD = 100 * y2, time = x, ID = ID)

  # Addition of a baseline variable.
  BL <- df[df$time == 0, c("ID", "PD")]
  names(BL) <- c("ID", "BL")
  df <- merge(df, BL)

  # Addition of change from baseline.
  df$Change <- df$PD - df$BL

  # Definition of treatment.
  df$trt <- ifelse(df$BL > mean(df$BL), "active", " placebo")

  return(df)
}

# Generate the data.
x <- make.data()

# ---
# Figures.

library(ggplot2)
gg <- ggplot(x, aes(x = time, y = Change, group = ID, color = ID))
gg <- gg + theme_bw()
gg <- gg + xlab("Time [h]") + ylab("Change from baseline")
gg <- gg + geom_line(linewidth = 1.1) + theme(legend.position = "none")
gg <- gg + facet_grid(. ~ trt)

# Addition of an auxiliary line at y=0.
gg2 <- gg + geom_hline(yintercept = 0, linewidth = 1.2)
```

```
# Symmetric y-axis limits.
gg3 <- gg2 + ylim(c(-1, 1) * max(abs(x$Change)))

# Arranging all plots into one figure.
library(cowplot)
plot_grid(gg, gg2, gg3, nrow = 1)
```

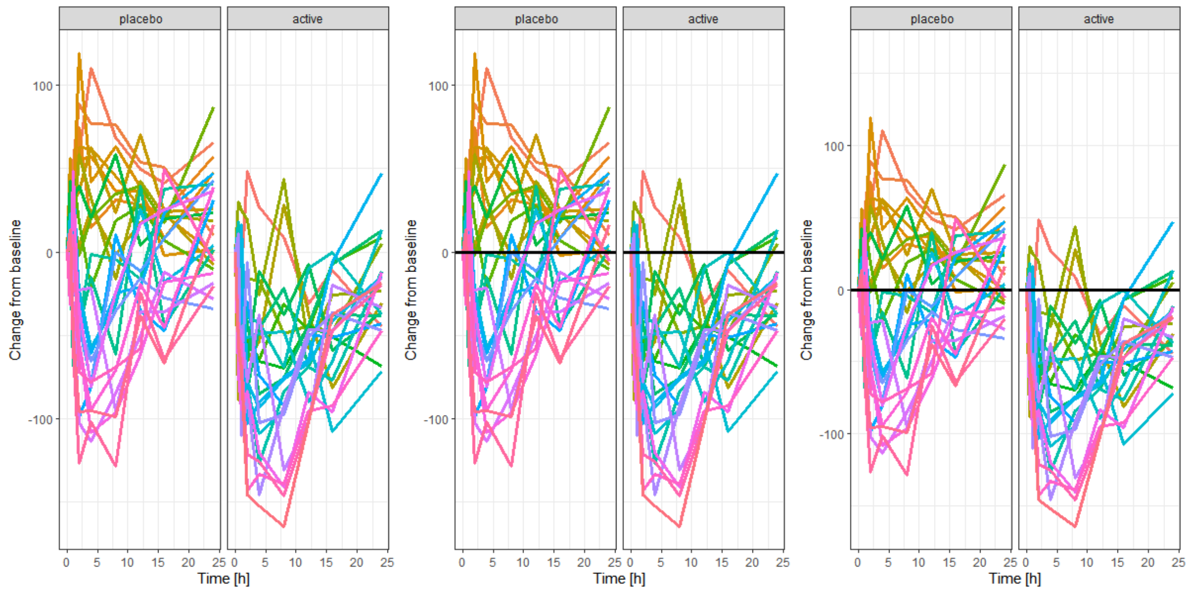

Figure 7: .

3

3 , .

: 10 3D . "X" . ( ) .
. 4 10, 20, 30, 40 .

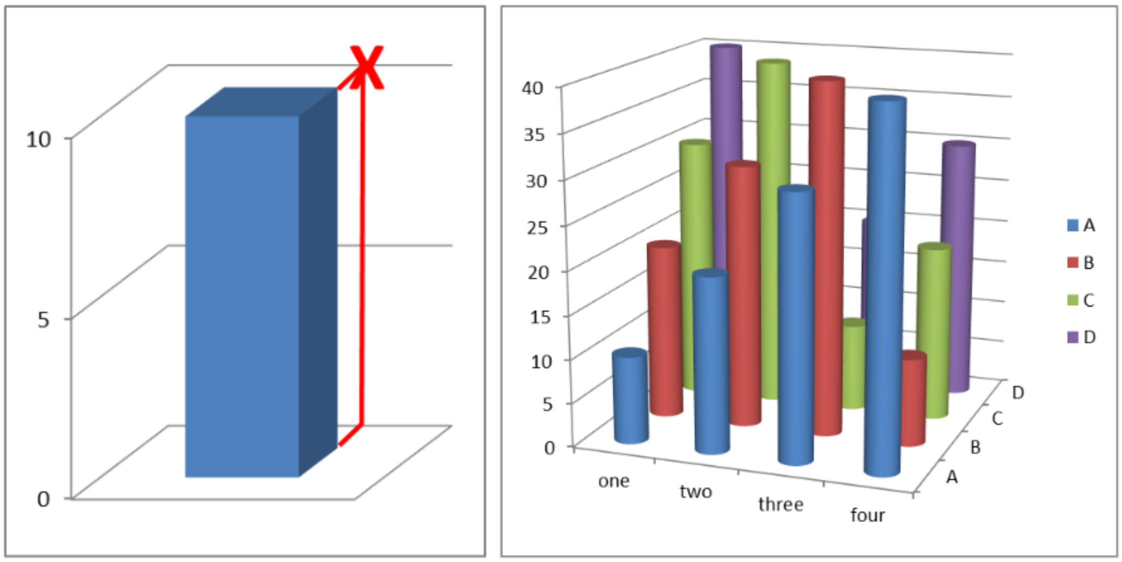

Figure 8: 3D .

. . .
(:), .
. .

, . . .

(,) (:). .

. .

(:) . , (:
) .

. . ,
.

. .

Few, Stephen. 2004. *Show Me the Numbers.* Burlingame, CA: Analytics Press.

Tufte, Edward R. 2001. *The Visual Display of Quantitative Information.* 2nd ed. Cheshire, CT: Graphics Press.