Universitätsmedizin
GREIFSWALD

Visualization with R:ggplot2

Marcus Vollmer

Institute of Bioinformatics, University Medicine Greifswald

// R-Developer-Meeting //

17 January 2019, Greifswald

Content

1 Introduction

2 Basic graphics building

3 Advanced examples

1. Introduction

ggplot2 package history

- Hadley Wickham implements `ggplot2` on the basis of **Leland Wilkinson** *The grammar of graphics* Springer (2005). ▸ PDF via OPAC which describes graphs using a formal notation

the **g**rammar of **g**raphics **plot**

- First release of `ggplot` on 28 October 2005 (version 0.2.2)
- First release of `ggplot2` on 10 June 2007 (version 0.5.1)
- Latest release of `ggplot2` on 25 October 2018 (version 3.1.0)

▸ ggplot2

ggplot2 release highlights

0.5.0	+ instead of functional style
0.6.0	documentation, auto legends
0.7.0–0.9.0	themes, facet_wrap, free scales, namespace, roxygen, S3, diaspora
1.0.0	new theme settings, brewer palettes, vertical justification for rotated text
2.0.0	official extension mechanism, new geoms, new default appearance, richer set of labelling options for facets, documentation has been overhauled
3.0.0	support of tidy evaluation (programmatically building plots in the same way you build data manipulation pipelines with dplyr), full support of simple features (GIS standard), stat function for aesthetics, tag, new functions for scales and guides
3.1.0	color = colour, new simple features functions and layers

Literature

Hadley Wickham *ggplot2: Elegant Graphics for Data Analysis* Springer-Verlag New York (2009). ▸ PDF via OPAC

Daniel Wollschläger *Grundlagen der Datenanalyse mit R: Eine anwendungsorientierte Einführung* Springer Spektrum (2017). ▸ PDF via OPAC

Online Tutorials
Garrett Grolemund and Hadley Wickham *R for Data Science* ▸ https://r4ds.had.co.nz/data-visualisation.html

Institute for Quantitative Social Science, Harvard University *R graphics with ggplot2 workshop notes* ▸ https://tutorials.iq.harvard.edu/R/Rgraphics

Statistik und ihre Anwendungen
Daniel Wollschläger
Grundlagen der Datenanalyse mit R
Eine anwendungsorientierte Einführung
4. Auflage
Springer Spektrum

Use R!
Hadley Wickham
ggplot2
Elegant Graphics for Data Analysis
Springer

Two-way plot production

`qplot()` quick plot

- uses some concepts of The Grammar of Graphics
- designed to be very similar to basic `plot()` function
- make it easy to produce basic graphs but may delay understanding philosophy of ggplot2

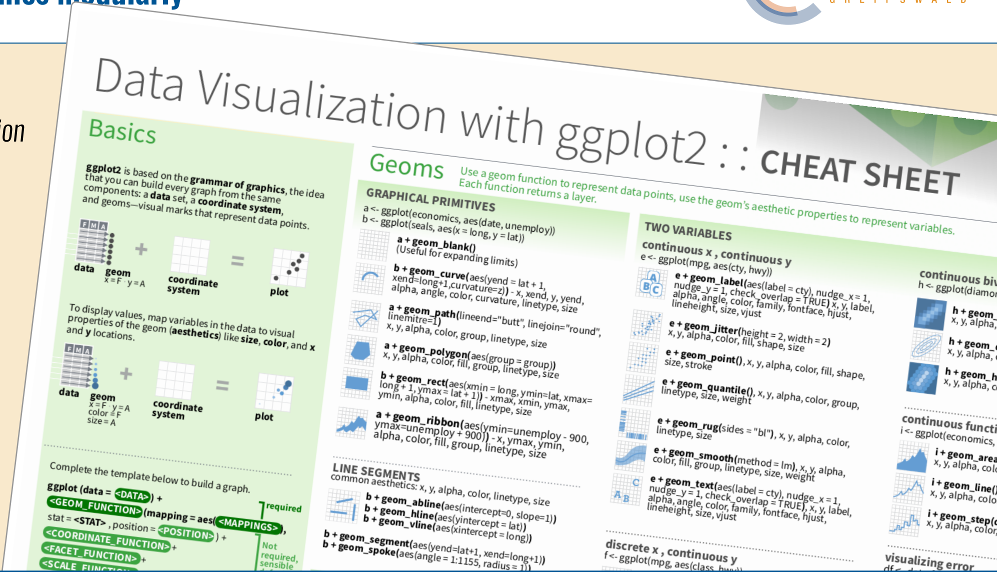
`ggplot()` grammar of graphics plot

- provides fuller implementation of The Grammar of Graphics
- allows much more flexibility when building graphs

Create graphics modularly

Data Visualization with ggplot2
RStudio

▸ Cheat Sheet



Data Visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = <STAT> , position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION
```

required

Not required, sensible

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

b + geom_segment(aes(yend=lat+1, xend=long+1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

TWO VARIABLES

continuous x , continuous y
e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**, x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y
f <- ggplot(mpg, aes(class, hwy))

continuous biv
h <- ggplot(diamon

h + geom_
x, y, alpha,

h + geom_
x, y, alpha,

h + geom_h
x, y, alpha, c

continuous functi
i <- ggplot(economics,

i + geom_area
x, y, alpha, colo

i + geom_line()
x, y, alpha, colo

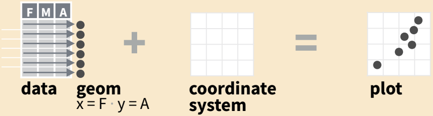
i + geom_step(
x, y, alpha, color,

visualizing error
df <- data

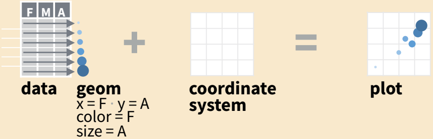
2. Basic graphics building

Basic components and syntax

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = **<STAT>** , position = **<POSITION>**) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

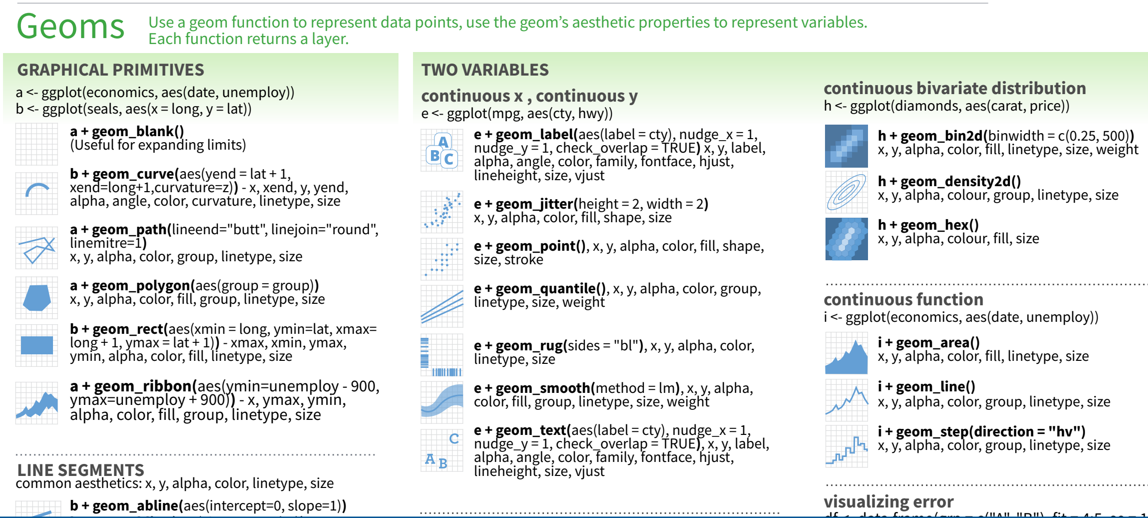
aesthetic mappings **data** **geom**

qplot(x = cty, y = hwy, data = mpg, geom = “point"**)** Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5’ x 5’ file named "plot.png" in working directory. Matches file type to file extension.

Basic components and syntax



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1)**)**

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**, x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

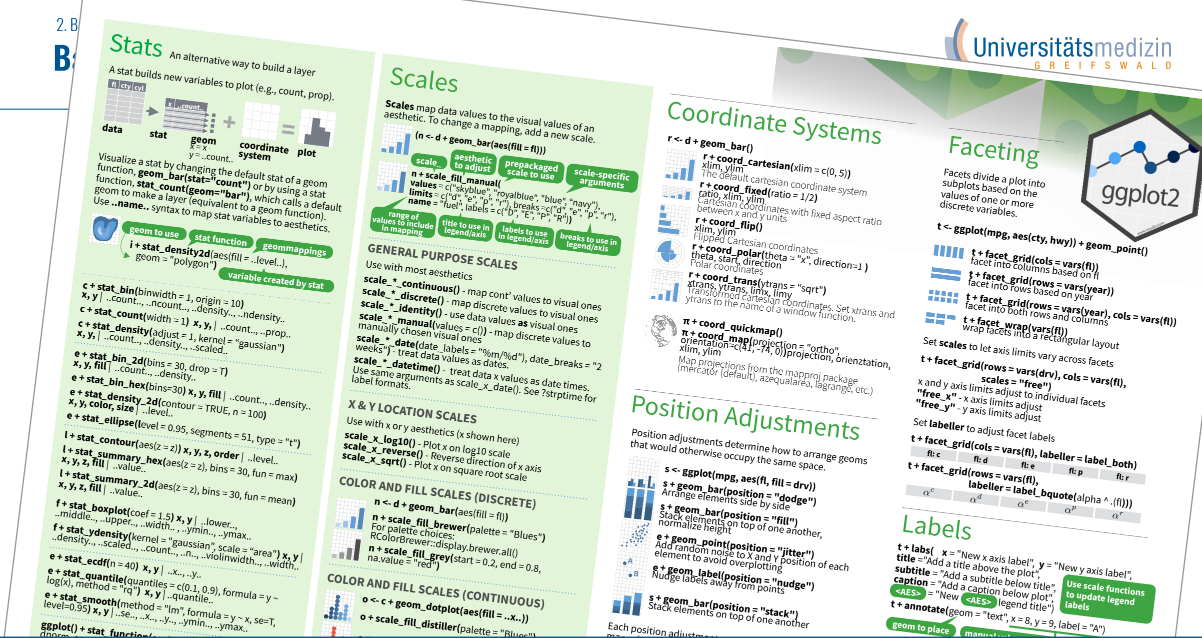
i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1

Ba



Stats An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).

Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1) **x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30) **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) **x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(

dnorm

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

(n <- d + geom_bar(aes(fill = fl)))

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

scale_
aesthetic to adjust
prepackaged scale to use
scale-specific arguments
range of values to include in mapping
title to use in legend/axis
labels to use in legend/axis
breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

scale_*_continuous() - map cont' values to visual ones
scale_*_discrete() - map discrete values to visual ones
scale_*_identity() - use data values **as** visual ones
scale_*_manual(values = c()**)** - map discrete values to manually chosen visual ones
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks"**)** - treat data values as dates.
scale_*_datetime() - treat data x values as date times. Use same arguments as scale_x_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

n <- d + geom_bar(aes(fill = fl)**)**

n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = ..x..))

o + scale_fill_distiller(palette = "Blues")

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
The default cartesian coordinate system

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction=1 **)**
theta, start, direction
Polar coordinates

r + coord_trans(ytrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0)**)**projection, orienztation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Arrange elements side by side

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height

e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting

e + geom_label(position = "nudge")
Nudge labels away from points

s + geom_bar(position = "stack")
Stack elements on top of one another

Each position adjustment

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(cols = vars(fl))
facet into columns based on fl

t + facet_grid(rows = vars(year))
facet into rows based on year

t + facet_grid(rows = vars(year), cols = vars(fl))
facet into both rows and columns

t + facet_wrap(vars(fl))
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free")
x and y axis limits adjust to individual facets
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(cols = vars(fl), labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(rows = vars(fl), labeller = label_bquote(alpha ^ .(fl)**))**

α^c	α^d	α^e	α^p	α^r

Labels

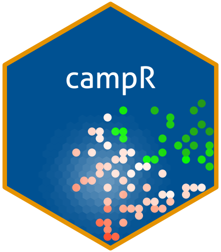
t + labs(**x** = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
<AES> = "New <AES> legend title")

t + annotate(geom = "text", x = 8, y = 9, label = "A")

Key components

`data`	data must be stored as an R data frame
`coordinate system`	describes 2D space that data is projected onto (Cartesian coordinates, polar coordinates, map projections, ...)
`geoms`	describes type of geometric objects that represent data
`aesthetics`	describes its visual characteristics (position, size, color, shape, ...)
`scales`	describes how visual characteristics are converted to displayed values (log scales, color scales, size scales, shape scales, ...)
`stats`	describes statistical transformations that typically summarize data (counts, means, medians, regression lines, ...)
`facets`	describes how data is split into subsets and displayed as multiple graphs
`annotations`	describes a geometric object to add text labels
`themes`	describes the look of non-data components of the plot

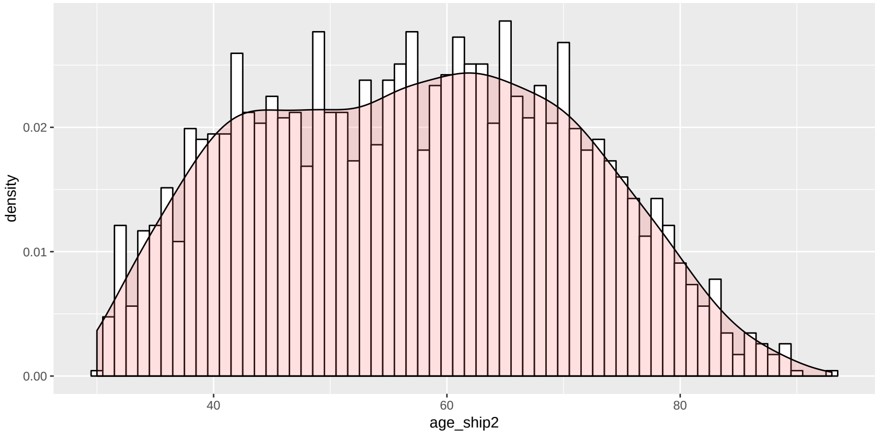
Practice



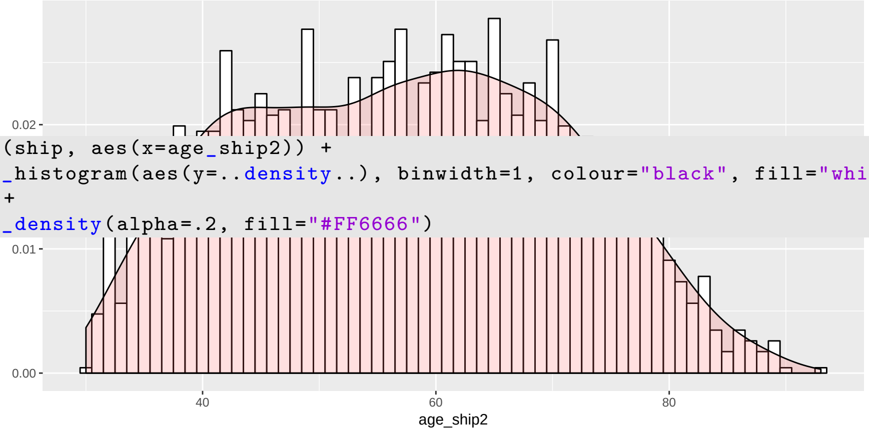
`https://github.com/MarcusVollmer/campR`

3. Advanced examples

Histogram with density

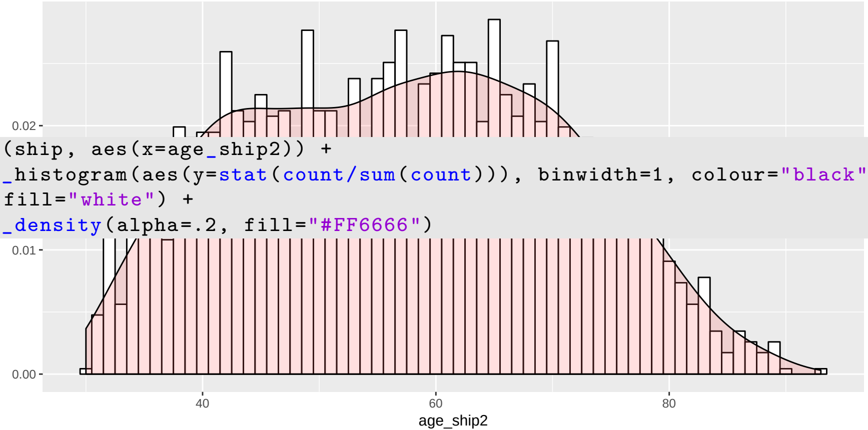


Histogram with density



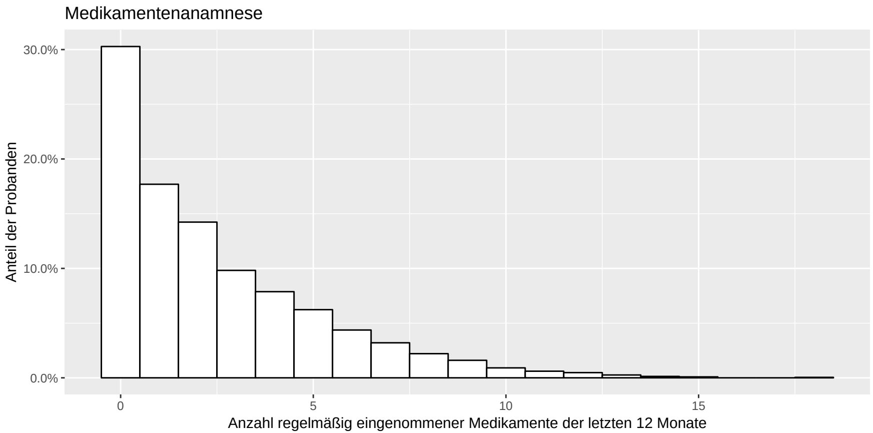
```
ggplot(ship, aes(x=age_ship2)) +
  geom_histogram(aes(y=..density..), binwidth=1, colour="black", fill="white")
      +
  geom_density(alpha=.2, fill="#FF6666")
```

Histogram with density

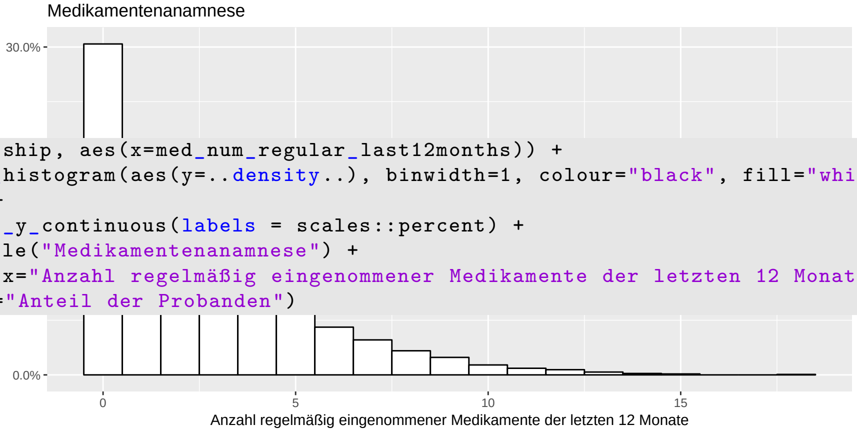


```
ggplot(ship, aes(x=age_ship2)) +
  geom_histogram(aes(y=stat(count/sum(count))), binwidth=1, colour="black",
      fill="white") +
  geom_density(alpha=.2, fill="#FF6666")
```

Percent scale

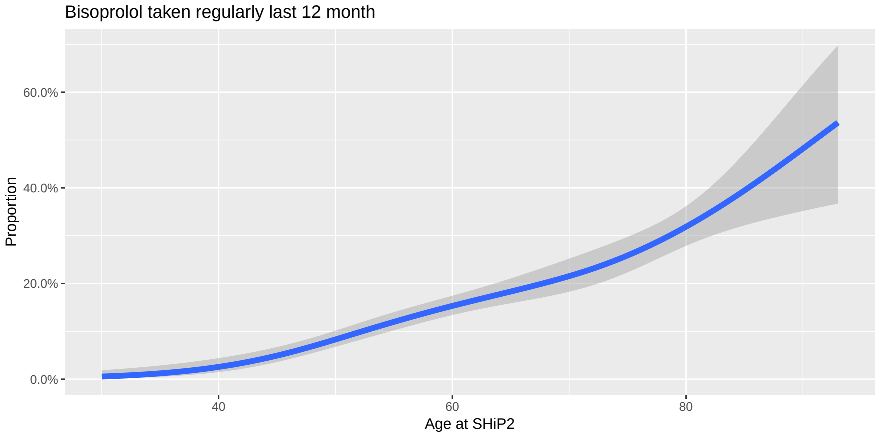


Percent scale

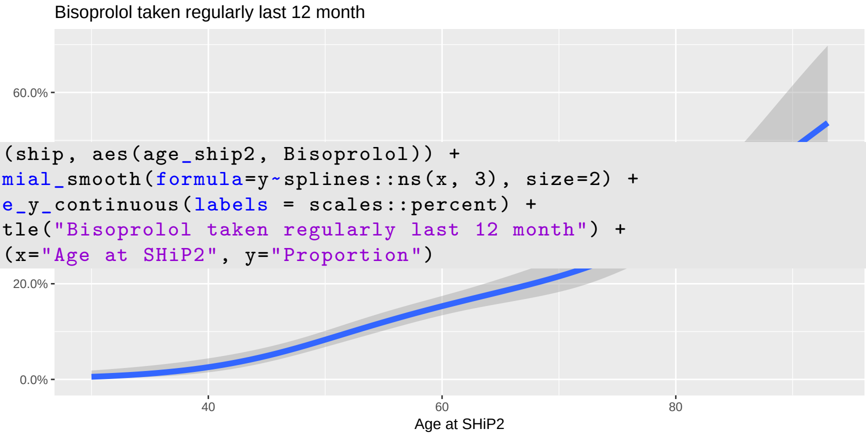


```
ggplot(ship, aes(x=med_num_regular_last12months)) +
  geom_histogram(aes(y=..density..), binwidth=1, colour="black", fill="white")
      +
  scale_y_continuous(labels = scales::percent) +
  ggtitle("Medikamentenanamnese") +
  labs(x="Anzahl regelmäßig eingenommener Medikamente der letzten 12 Monate", y
      ="Anteil der Probanden")
```

Binomial smooth

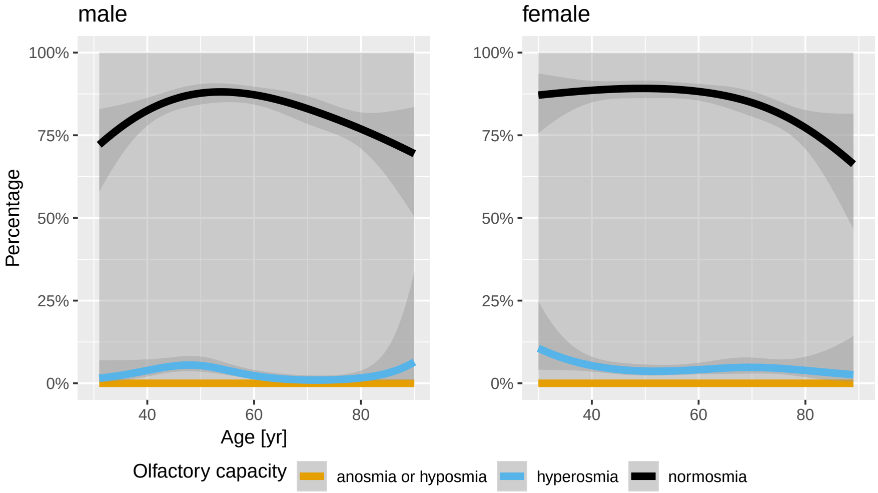


Binomial smooth

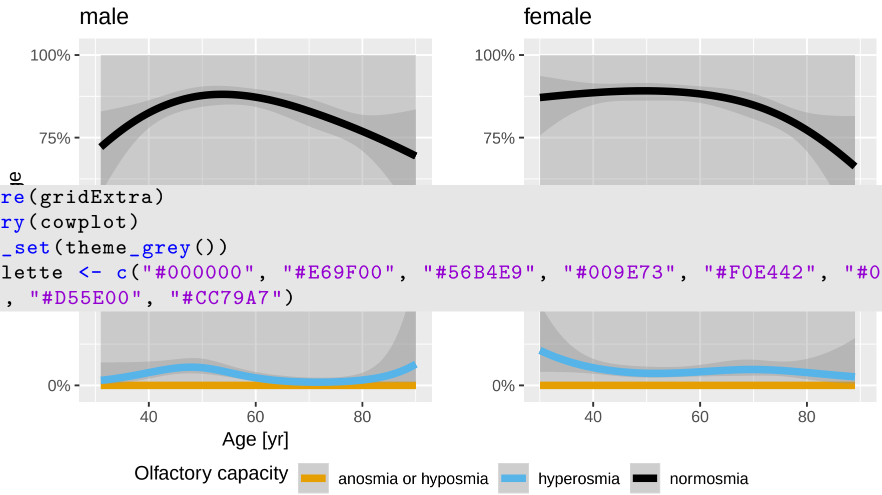


```
ggplot(ship, aes(age_ship2, Bisoprolol)) +
  binomial_smooth(formula=y~splines::ns(x, 3), size=2) +
  scale_y_continuous(labels = scales::percent) +
  ggtitle("Bisoprolol taken regularly last 12 month") +
  labs(x="Age at SHiP2", y="Proportion")
```

Shared legend, manual colors



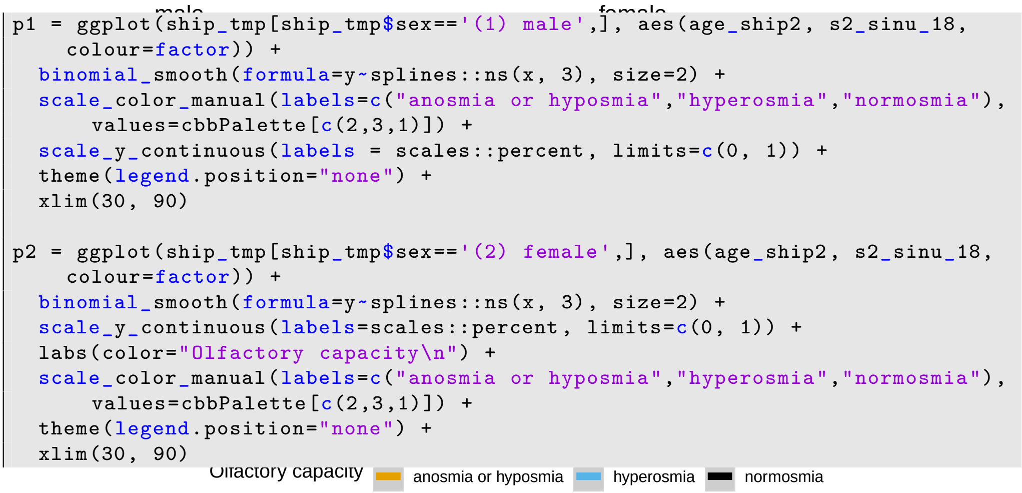
Shared legend, manual colors



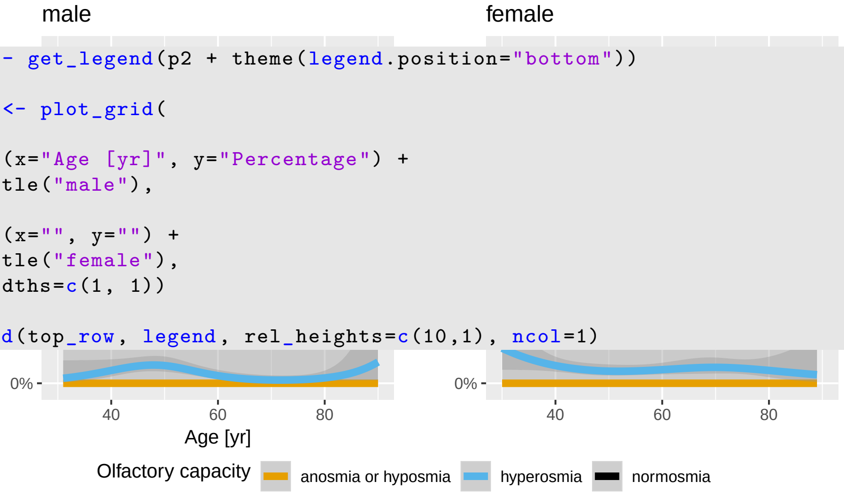
```
require(gridExtra)
library(cowplot)
theme_set(theme_grey())
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2
    ", "#D55E00", "#CC79A7")
```

Shared legend, manual colors

```
p1 = ggplot(ship_tmp[ship_tmp$sex=='(1) male',], aes(age_ship2, s2_sinu_18,
     colour=factor)) +
  binomial_smooth(formula=y~splines::ns(x, 3), size=2) +
  scale_color_manual(labels=c("anosmia or hyposmia","hyperosmia","normosmia"),
      values=cbbPalette[c(2,3,1)]) +
  scale_y_continuous(labels = scales::percent, limits=c(0, 1)) +
  theme(legend.position="none") +
  xlim(30, 90)

p2 = ggplot(ship_tmp[ship_tmp$sex=='(2) female',], aes(age_ship2, s2_sinu_18,
     colour=factor)) +
  binomial_smooth(formula=y~splines::ns(x, 3), size=2) +
  scale_y_continuous(labels=scales::percent, limits=c(0, 1)) +
  labs(color="Olfactory capacity\n") +
  scale_color_manual(labels=c("anosmia or hyposmia","hyperosmia","normosmia"),
      values=cbbPalette[c(2,3,1)]) +
  theme(legend.position="none") +
  xlim(30, 90)
```

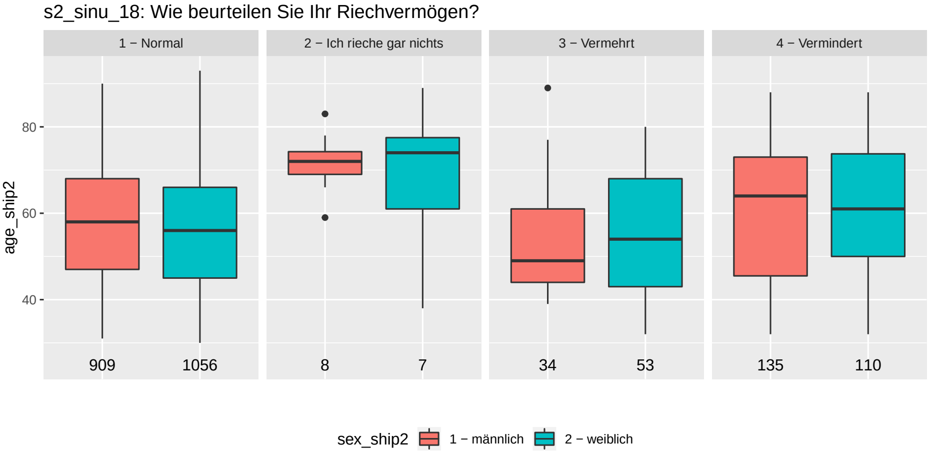


Shared legend, manual colors

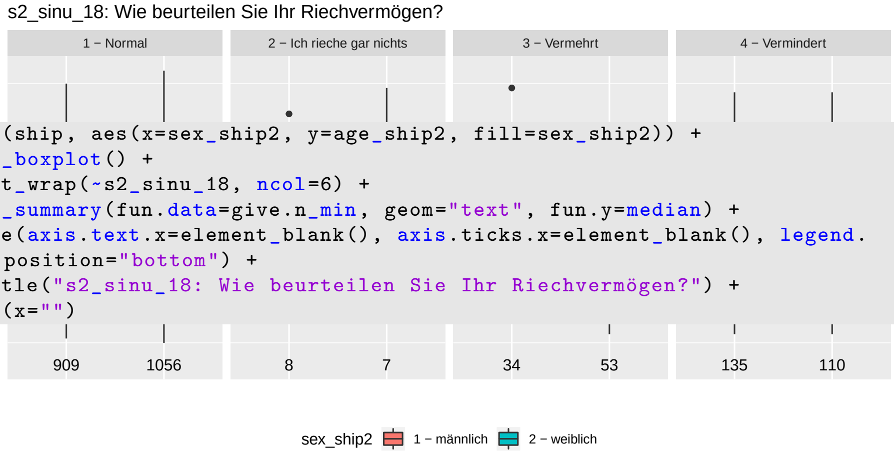


```
legend <- get_legend(p2 + theme(legend.position="bottom"))

top_row <- plot_grid(
  p1 +
    labs(x="Age [yr]", y="Percentage") +
    ggtitle("male"),
  p2 +
    labs(x="", y="") +
    ggtitle("female"),
  rel_widths=c(1, 1))

plot_grid(top_row, legend, rel_heights=c(10,1), ncol=1)
```

Grouped boxplots in facets with sample sizes

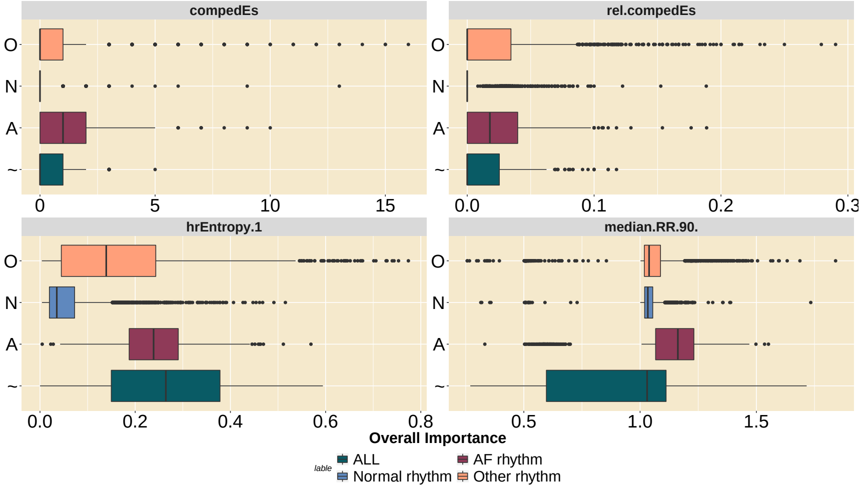


Grouped boxplots in facets with sample sizes

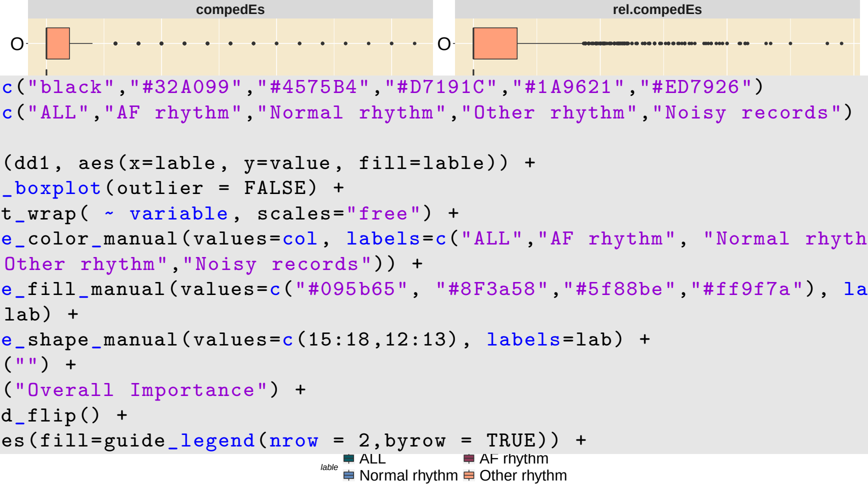


```r
ggplot(ship, aes(x=sex_ship2, y=age_ship2, fill=sex_ship2)) +
  geom_boxplot() +
  facet_wrap(~s2_sinu_18, ncol=6) +
  stat_summary(fun.data=give.n_min, geom="text", fun.y=median) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank(), legend.
      position="bottom") +
  ggtitle("s2_sinu_18: Wie beurteilen Sie Ihr Riechvermögen?") +
  labs(x="")
```

Horizontal boxplots in facets with color theme

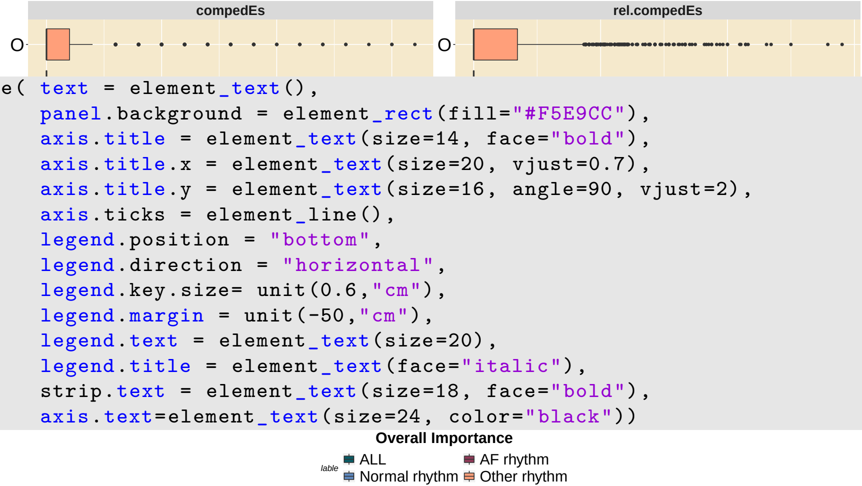


Horizontal boxplots in facets with color theme



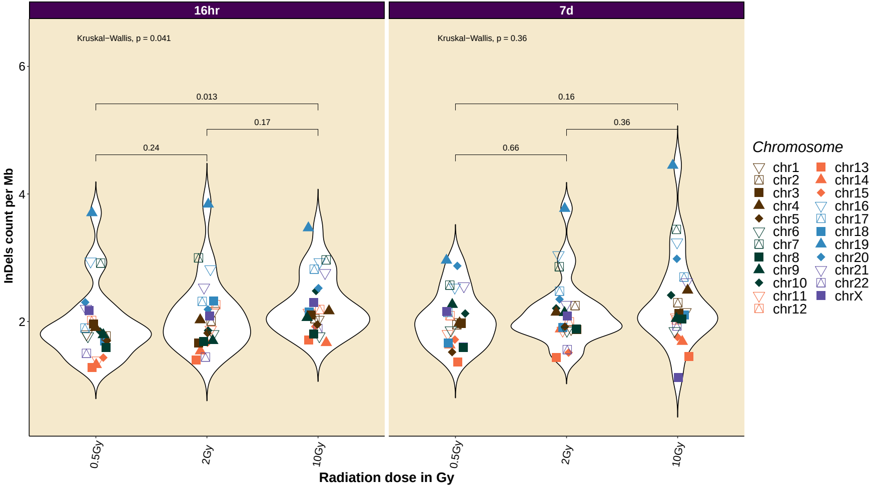
```
col = c("black","#32A099","#4575B4","#D7191C","#1A9621","#ED7926")
lab = c("ALL","AF rhythm","Normal rhythm","Other rhythm","Noisy records")

ggplot(dd1, aes(x=lable, y=value, fill=lable)) +
  geom_boxplot(outlier = FALSE) +
  facet_wrap( ~ variable, scales="free") +
  scale_color_manual(values=col, labels=c("ALL","AF rhythm", "Normal rhythm","
      Other rhythm","Noisy records")) +
  scale_fill_manual(values=c("#095b65", "#8F3a58","#5f88be","#ff9f7a"), labels=
      lab) +
  scale_shape_manual(values=c(15:18,12:13), labels=lab) +
  xlab("") +
  ylab("Overall Importance") +
  coord_flip() +
  guides(fill=guide_legend(nrow = 2,byrow = TRUE)) +
```

Horizontal boxplots in facets with color theme

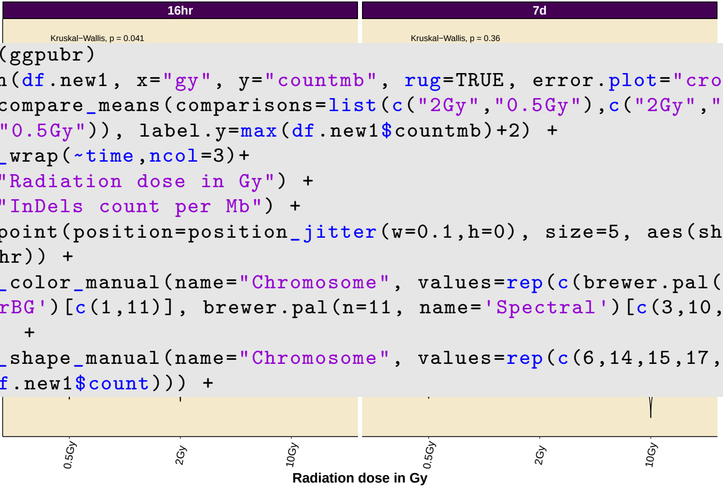


```
theme( text = element_text(),
       panel.background = element_rect(fill="#F5E9CC"),
       axis.title = element_text(size=14, face="bold"),
       axis.title.x = element_text(size=20, vjust=0.7),
       axis.title.y = element_text(size=16, angle=90, vjust=2),
       axis.ticks = element_line(),
       legend.position = "bottom",
       legend.direction = "horizontal",
       legend.key.size= unit(0.6,"cm"),
       legend.margin = unit(-50,"cm"),
       legend.text = element_text(size=20),
       legend.title = element_text(face="italic"),
       strip.text = element_text(size=18, face="bold"),
       axis.text=element_text(size=24, color="black"))
```

Violin plots in facets with statistics

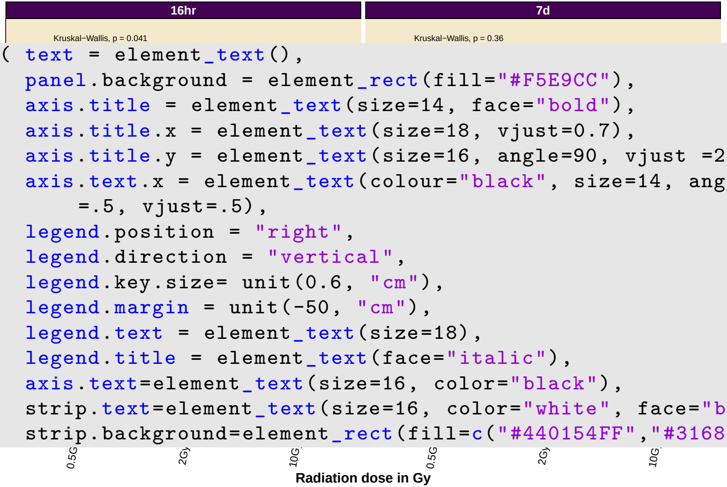


Violin plots in facets with statistics



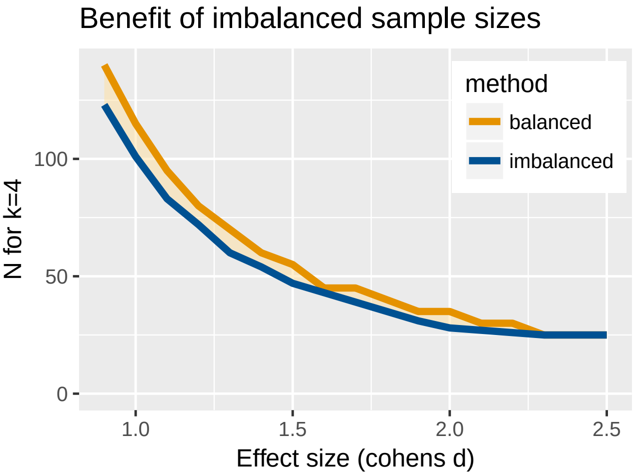
```
library(ggpubr)
ggviolin(df.new1, x="gy", y="countmb", rug=TRUE, error.plot="crossbar")+
  stat_compare_means(comparisons=list(c("2Gy","0.5Gy"),c("2Gy","10Gy"),c("10Gy"
      ,"0.5Gy")), label.y=max(df.new1$countmb)+2) +
  facet_wrap(~time,ncol=3)+
  xlab("Radiation dose in Gy") +
  ylab("InDels count per Mb") +
  geom_point(position=position_jitter(w=0.1,h=0), size=5, aes(shape=chr,color=
      chr)) +
  scale_color_manual(name="Chromosome", values=rep(c(brewer.pal(n=11, name='
      BrBG')[c(1,11)], brewer.pal(n=11, name='Spectral')[c(3,10,11)]), each=5)
      )  +
  scale_shape_manual(name="Chromosome", values=rep(c(6,14,15,17,18), 5*length(
      df.new1$count))) +
```

Violin plots in facets with statistics

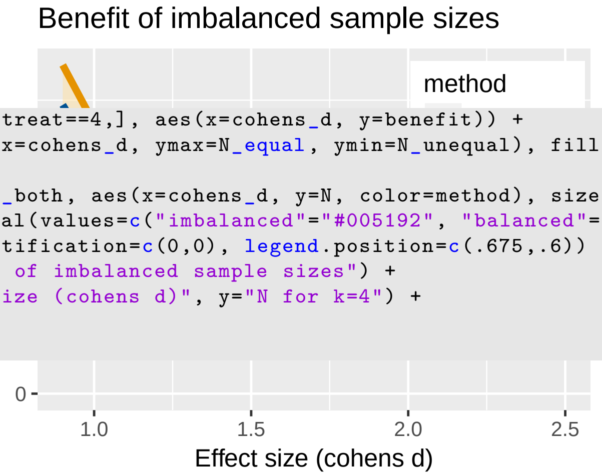


```
theme( text = element_text(),
       panel.background = element_rect(fill="#F5E9CC"),
       axis.title = element_text(size=14, face="bold"),
       axis.title.x = element_text(size=18, vjust=0.7),
       axis.title.y = element_text(size=16, angle=90, vjust =2),
       axis.text.x = element_text(colour="black", size=14, angle=80, hjust
           =.5, vjust=.5),
       legend.position = "right",
       legend.direction = "vertical",
       legend.key.size= unit(0.6, "cm"),
       legend.margin = unit(-50, "cm"),
       legend.text = element_text(size=18),
       legend.title = element_text(face="italic"),
       axis.text=element_text(size=16, color="black"),
       strip.text=element_text(size=16, color="white", face="bold"),
       strip.background=element_rect(fill=c("#440154FF","#31688EFF")))
```

geom_ribbon()

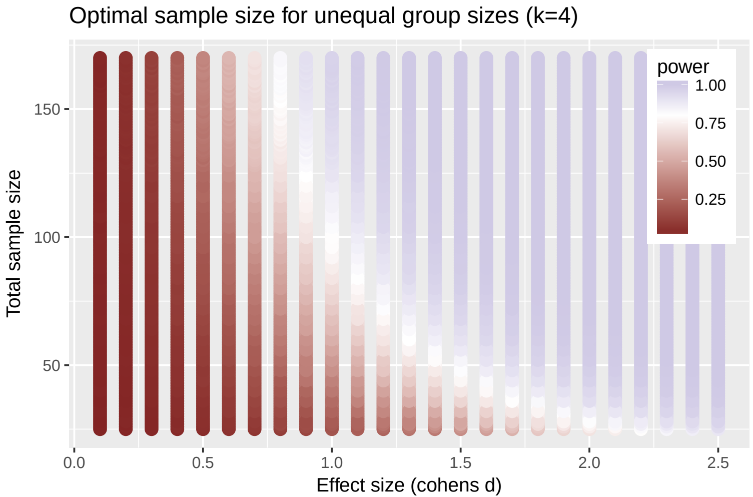

Benefit of imbalanced sample sizes
N for k=4
100
50
0
1.0
1.5
2.0
2.5
Effect size (cohens d)
method
balanced
imbalanced

geom_ribbon()

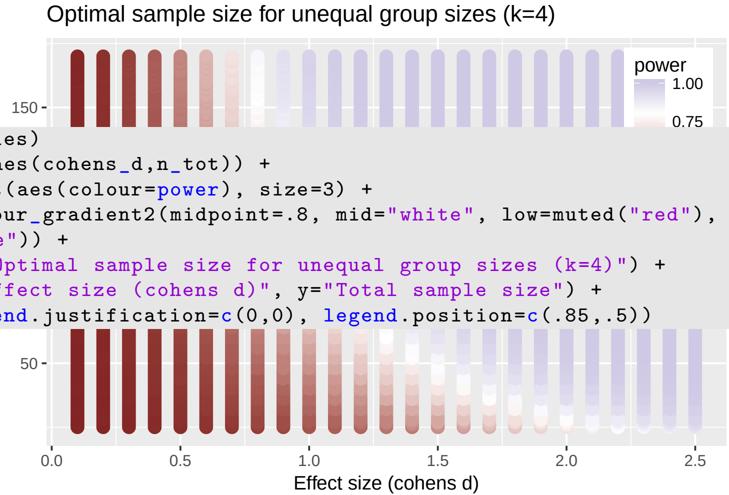


```r
ggplot(data=S[S$n_treat==4,], aes(x=cohens_d, y=benefit)) +
  geom_ribbon(aes(x=cohens_d, ymax=N_equal, ymin=N_unequal), fill='#FFE1A1',
      alpha=.5) +
  geom_line(data=N_both, aes(x=cohens_d, y=N, color=method), size=1.5) +
  scale_color_manual(values=c("imbalanced"="#005192", "balanced"="#E59200")) +
  theme(legend.justification=c(0,0), legend.position=c(.675,.6)) +
  ggtitle("Benefit of imbalanced sample sizes") +
  labs(x="Effect size (cohens d)", y="N for k=4") +
  xlim(.9,2.5) +
  ylim(0,140)
```

scale_colour_gradient2()



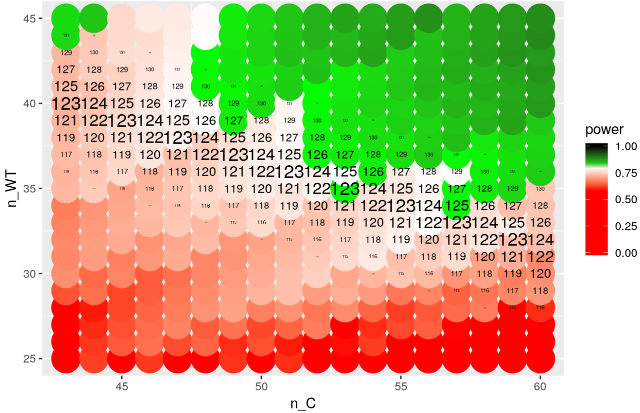
scale_colour_gradient2()



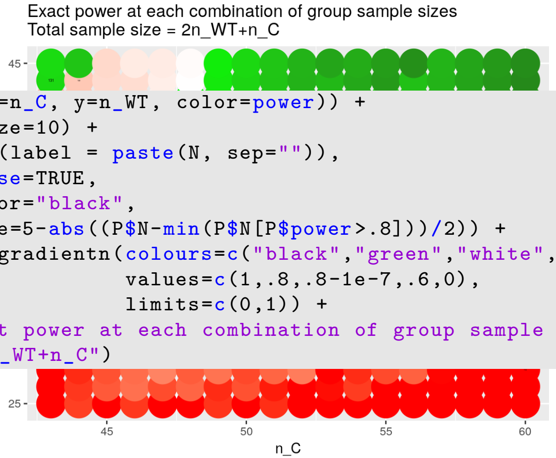
```
library(scales)
ggplot(t2, aes(cohens_d,n_tot)) +
  geom_point(aes(colour=power), size=3) +
  scale_colour_gradient2(midpoint=.8, mid="white", low=muted("red"), high=muted
       ("blue")) +
  ggtitle("Optimal sample size for unequal group sizes (k=4)") +
  labs(x="Effect size (cohens d)", y="Total sample size") +
  theme(legend.justification=c(0,0), legend.position=c(.85,.5))
```

scale_colour_gradientn() and geom_text()

Exact power at each combination of group sample sizes
Total sample size = 2n_WT+n_C

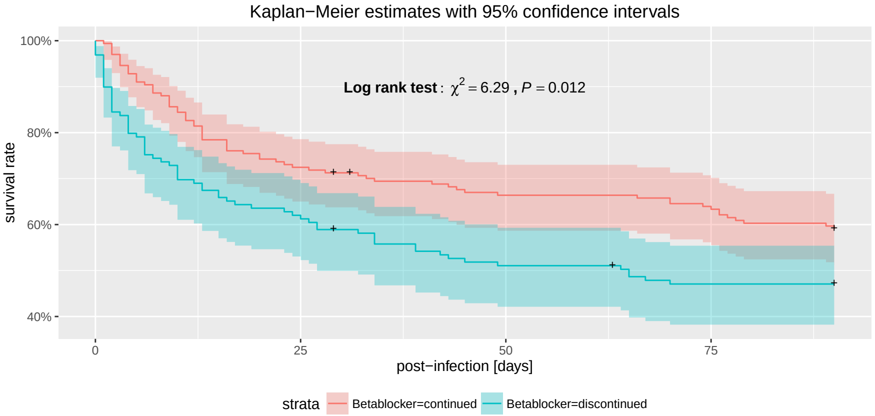


scale_colour_gradientn() and geom_text()



```
ggplot(P, aes(x=n_C, y=n_WT, color=power)) +
  geom_point(size=10) +
  geom_text(aes(label = paste(N, sep="")),
            parse=TRUE,
            color="black",
            size=5-abs((P$N-min(P$N[P$power>.8]))/2)) +
  scale_colour_gradientn(colours=c("black","green","white","red","red"),
                         values=c(1,.8,.8-1e-7,.6,0),
                         limits=c(0,1)) +
  ggtitle("Exact power at each combination of group sample sizes\nTotal sample
      size = 2n_WT+n_C")
```

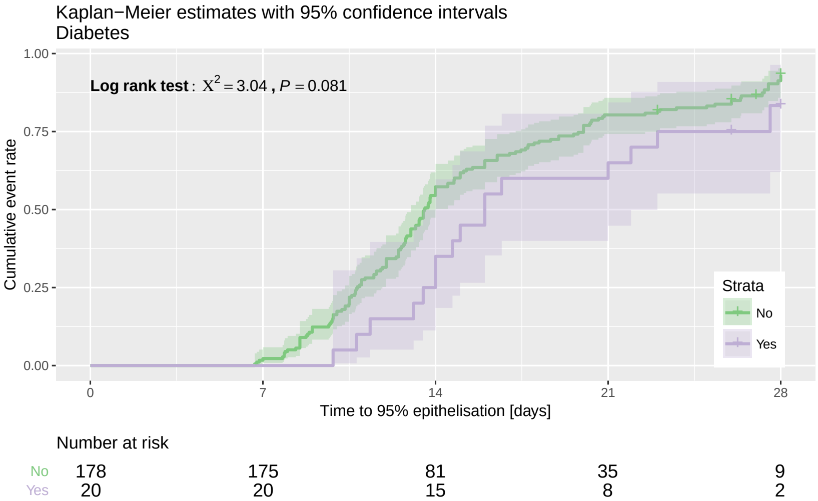
Kaplan-Meier plot with logrank result



Kaplan-Meier plot with logrank result

```r
library(survival)
S = Surv(survival_days90, survival_event90)
km = survfit(S ~ Betablocker, data=ds5b, conf.type="log-log")

# Log-rank test
survd = survdiff(S ~ Betablocker, data=ds5b, rho=0)
tmp = paste("bold(Log~rank~test:~chi^2", "==", round(survd$chisq,2), "~','", "~italic('P')==", round(signif(1 - pchisq(survd$chisq, 1)),3), ")")

library(ggfortify)
myColors = c(rgb(55,142,0,maxColorValue=255), rgb(0,107,250,maxColorValue=255))
ggplot2::autoplot(km,
                  xlab="post-infection [days]",
                  ylab="survival rate",
                  main="Kaplan-Meier estimates with 95% confidence intervals")
                      +
  theme(legend.position="bottom", legend.direction="horizontal") +
  annotate("text", 45, 0.9, parse=T, label=tmp) +
  scale_color_manual(values=myColors)
```

Kaplan-Meier estimator with number-at-risk table



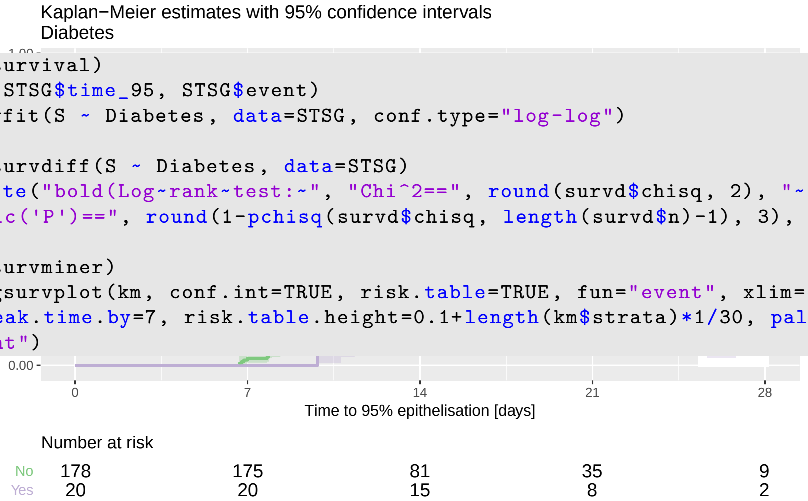
Kaplan−Meier estimates with 95% confidence intervals
Diabetes

Log rank test: $\mathrm{X}^2 = 3.04$ **,** $P = 0.081$

Number at risk

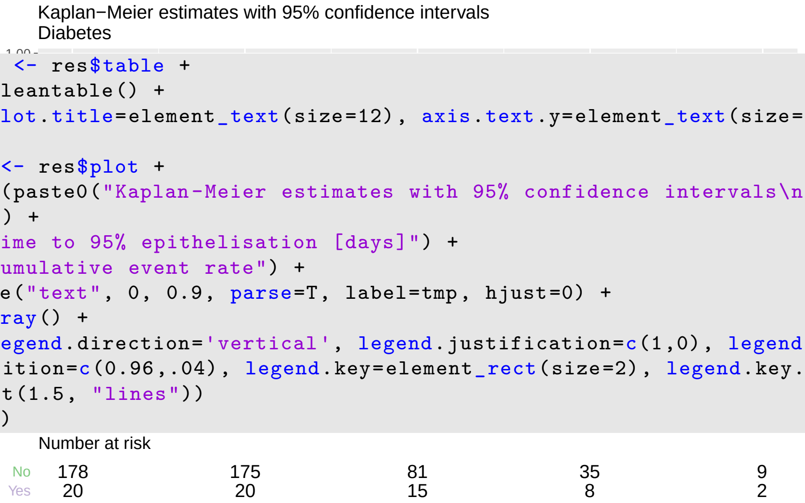
No	178	175	81	35	9
Yes	20	20	15	8	2

Kaplan-Meier estimator with number-at-risk table



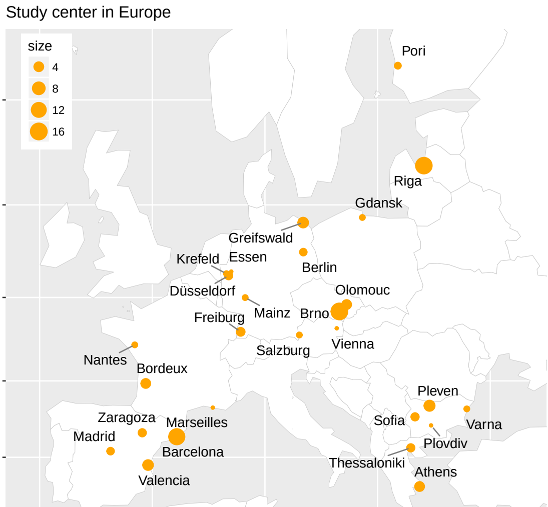
```r
library(survival)
S = Surv(STSG$time_95, STSG$event)
km = survfit(S ~ Diabetes, data=STSG, conf.type="log-log")

survd = survdiff(S ~ Diabetes, data=STSG)
tmp = paste("bold(Log~rank~test:~", "Chi^2==", round(survd$chisq, 2), "~','","~
    italic('P')==", round(1-pchisq(survd$chisq, length(survd$n)-1), 3), ")")

library(survminer)
res <- ggsurvplot(km, conf.int=TRUE, risk.table=TRUE, fun="event", xlim=c(0,28)
    , break.time.by=7, risk.table.height=0.1+length(km$strata)*1/30, palette="
    Accent")
```

Kaplan-Meier estimator with number-at-risk table



```r
res$table <- res$table +
  theme_cleantable() +
  theme(plot.title=element_text(size=12), axis.text.y=element_text(size=10))

res$plot <- res$plot +
  ggtitle(paste0("Kaplan-Meier estimates with 95% confidence intervals\n",vars[
      i])) +
  xlab("Time to 95% epithelisation [days]") +
  ylab("Cumulative event rate") +
  annotate("text", 0, 0.9, parse=T, label=tmp, hjust=0) +
  theme_gray() +
  theme(legend.direction='vertical', legend.justification=c(1,0), legend.
      position=c(0.96,.04), legend.key=element_rect(size=2), legend.key.size=
      unit(1.5, "lines"))
print(res)
```

Study center size on ggmap



Study center size on ggmap

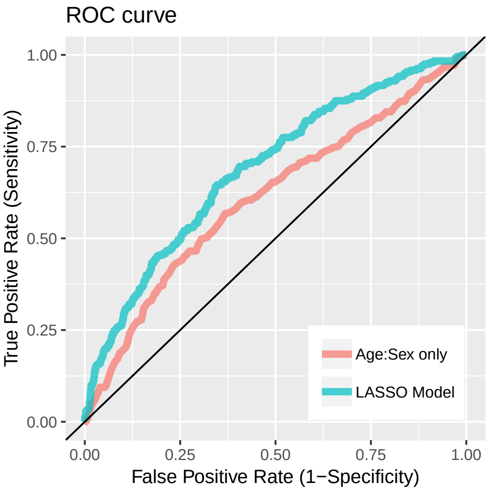
```
library(ggmap)
library(rworldmap)
library(ggrepel)

Center = geocode(paste(unique(STSG$Location), "Europe"), source="google")
Center$city =unique(STSG$Location)
Center = Center[order(Center$city),]
Center$size = as.double(table(STSG$Location)/2)

# Get the world map
worldMap <- getMap()
All = 1:length(worldMap$NAME)

# Extract longitude and latitude border's coordinates of all states
Coords = lapply(All, function(i){
  df <- data.frame(worldMap@polygons[[i]]@Polygons[[1]]@coords)
  df$region = as.character(worldMap$NAME[i])
  colnames(df) = list("long", "lat", "region")
  return(df)
})
Coords <- do.call("rbind", Coords)
```

Study center size on ggmap

Study center in Europe

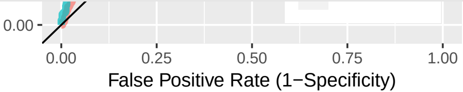
```
ggplot() +
  geom_polygon(data=Coords, aes(x=long, y=lat, group=region), colour='grey80',
      size=0.1, fill="white") +
  coord_map(xlim=c(-13, 35), ylim=c(35, 63)) +
  geom_point(data=Center, aes(x=lon, y=lat, size=size), color="orange") +
  geom_text_repel(data=Center, aes(x=lon, y=lat, label=city),
                  color='black',
                  box.padding=unit(0.25, "lines"),
                  point.padding=unit(0.3, "lines"),
                  segment.color='grey50') +
  ggtitle("Study center in Europe") +
  theme(axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        axis.text.x=element_blank(),
        axis.text.y=element_blank(),
        legend.position=c(.075, .875))
```

ROC curve


ROC curve
True Positive Rate (Sensitivity)
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
False Positive Rate (1−Specificity)
Age:Sex only
LASSO Model

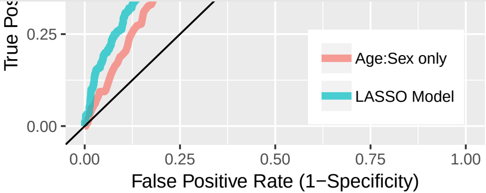
ROC curve

```
library(ROCR)
pred = predict(mymodel_reduced_o, newdata=subset(test1, type="response"))
pr = prediction(pred, test1$s2_sinu_18_o)
prf1 = performance(pr, "tpr", "fpr")

test1 = test[complete.cases(test[,c("age_ship2","sex")]),]
pred = predict(mymodel_reduced2_o, newdata=subset(test1, type="response"))
pr = prediction(pred, test1$s2_sinu_18_o)
prf2 = performance(pr, "tpr", "fpr")

# Plot ROC curve
df = data.frame(c(prf1@x.values[[1]], prf2@x.values[[1]]))
df$TPR = c(prf1@y.values[[1]], prf2@y.values[[1]])
colnames(df) = c("FPR", "TPR")
df$Label = factor(c(rep("LASSO Model", NROW(prf1@x.values[[1]])),
                    rep("Age:Sex only", NROW(prf2@x.values[[1]]))))
```

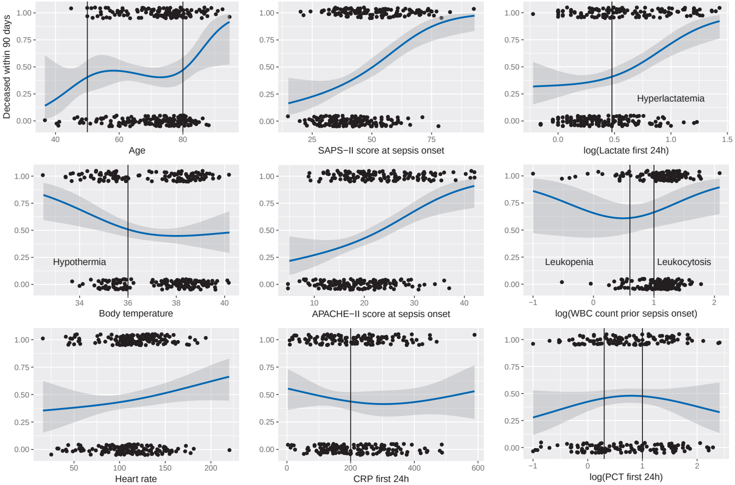


ROC curve

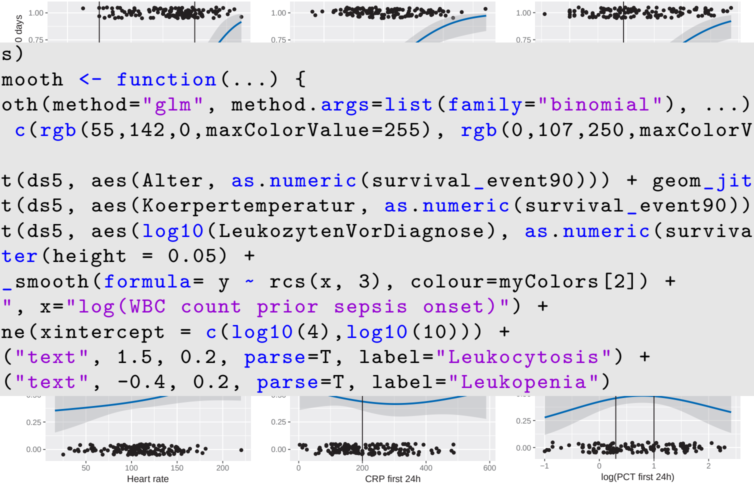
```
ggplot(df, aes(FPR, TPR, color=Label)) +
  geom_line(size=2, alpha=0.7) +
  labs(title="ROC curve",
       x="False Positive Rate (1-Specificity)",
       y="True Positive Rate (Sensitivity)") +
  geom_abline(intercept=0, slope=1) +
  theme(legend.direction="vertical",
        legend.justification=c(1,0),
        legend.position=c(0.95, 0.05),
        legend.key=element_rect(size=2),
        legend.key.size=unit(1.5, "lines")) +
  theme(legend.title=element_blank())
```



Multiplot with splines and geom_jitter

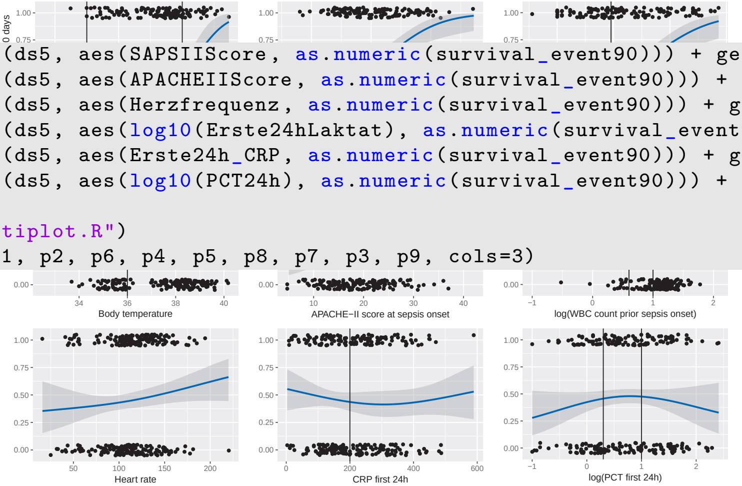


Multiplot with splines and geom_jitter



```
require(rms)
binomial_smooth <- function(...) {
  geom_smooth(method="glm", method.args=list(family="binomial"), ...)}
myColors = c(rgb(55,142,0,maxColorValue=255), rgb(0,107,250,maxColorValue=255))

p1 = ggplot(ds5, aes(Alter, as.numeric(survival_event90))) + geom_jitter(height = 0.05)
p2 = ggplot(ds5, aes(Koerpertemperatur, as.numeric(survival_event90))) + geom_jitter(he
p3 = ggplot(ds5, aes(log10(LeukozytenVorDiagnose), as.numeric(survival_event90))) +
  geom_jitter(height = 0.05) +
  binomial_smooth(formula= y ~ rcs(x, 3), colour=myColors[2]) +
  labs(y="", x="log(WBC count prior sepsis onset)") +
  geom_vline(xintercept = c(log10(4),log10(10))) +
  annotate("text", 1.5, 0.2, parse=T, label="Leukocytosis") +
  annotate("text", -0.4, 0.2, parse=T, label="Leukopenia")
```

Multiplot with splines and geom_jitter



```
p4 = ggplot(ds5, aes(SAPSIIScore, as.numeric(survival_event90))) + geom_jitter(height =
p5 = ggplot(ds5, aes(APACHEIIScore, as.numeric(survival_event90))) + geom_jitter(height
p6 = ggplot(ds5, aes(Herzfrequenz, as.numeric(survival_event90))) + geom_jitter(height
p7 = ggplot(ds5, aes(log10(Erste24hLaktat), as.numeric(survival_event90))) + geom_jitte
p8 = ggplot(ds5, aes(Erste24h_CRP, as.numeric(survival_event90))) + geom_jitter(height
p9 = ggplot(ds5, aes(log10(PCT24h), as.numeric(survival_event90))) + geom_jitter(height

source("multiplot.R")
multiplot(p1, p2, p6, p4, p5, p8, p7, p3, p9, cols=3)
```

Thank You for Your Attention!

Universitätsmedizin GREIFSWALD

4. Appendix

Copyright notices

Sources

https://qz.com/1007328/all-hail-ggplot2-the-code-powering-all-those-excellent-charts-is-10-years-old/
https://www.hse.ru/data/2015/04/24/1095313390/ggplot_intro.pdf
https://learnr.wordpress.com/2009/08/26/ggplot2-version-of-figures-in-lattice-multivariate-data-visualization-with-r-final-part/
https://vita.had.co.nz/papers/future-ig.pdf
https://blog.rstudio.com/2015/12/21/ggplot2-2-0-0/
https://www.tidyverse.org/articles/2018/07/ggplot2-3-0-0/
https://resources.rstudio.com/articles-on-tidyverse/ggplot2-3-1-0