Notes on using R

Pedro J. Aphalo

Git: tag ‘none’, committed with hash ‘none’ on ‘none’

Contents

Preface

This series of Notes cover different aspects of the use of R. They are meant to be use as a complement to a course or book, as explanations are short and terse. We do not discuss here statistics, just R as a tool and language for data manipulation and display. The idea is a bit like how children learn a language: they work-out what the rules are simply by listening to people speak. I do give some explanations and comments, but the idea of this notes is mainly for you to use the numerous examples to find-out by yourself the overall patterns and coding philosophy behind the R language.

This is work-in-progress. I will appreciate suggestions for further examples, notification of errors and unclear things and any bigger contributions. Many of the examples here have been collected from diverse sources over many years and because of this not all sources are acknowledged. If you recognize any example as yours or someone else's please let me know so that I can add a proper acknowledgement.

1 Plots with 'ggplot', 'ggrepel' and 'ggpmisc'

1.1 Packages used in this chapter

For executing the examples listed in this chapter you need first to load the following packages from the library:

```
library(ggplot2)
library(ggrepel)
library(ggpmisc)
library(xts)
```

We set a font larger size than the default

```
theme_set(theme_grey(16))
```

1.2 'ggpmisc'

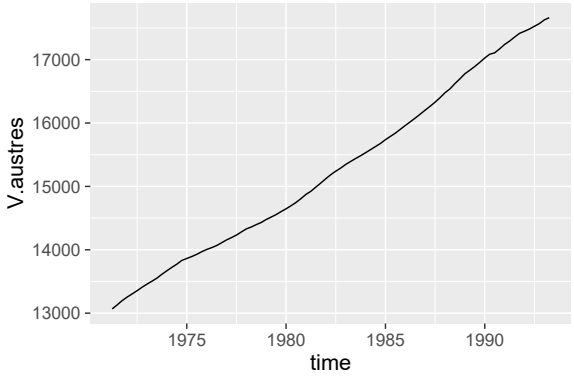
Package 'ggpmisc' is a small package developed by myself as a result of questions from work mates and in Stackoverflow, or functionality that I have needed in my own research or for teaching.

1.2.1 Plotting time-series

Instead of creating a new statistics or geometry for plotting time series we provide a function that can be used to convert time series objects into data frames suitable for plotting with 'ggplot2'. A single function `try_data_frame()` accepts time series objects saved with different packages as well as R's native `ts` objects. The *magic* is done mainly by package 'xts' to which we add a very simple wrapper to obtain a data frame.

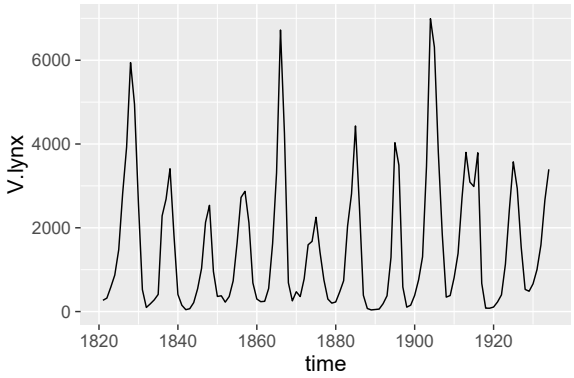
We exemplify this with some of the time series data included in R. In the first example we use the default format for time.

```
ggplot(try_data_frame(austres),
       aes(time, V.austres)) +
  geom_line()
```



In the second example we use years in numeric format for expressing 'time'.

```
ggplot(try_data_frame(lynx, "year", as.numeric = TRUE),
       aes(x = time, y = V.lynx)) +
  geom_line()
```



Multivariate time series are also supported.

1.2.2 New stats

Package 'ggpmisc' provides new stats: `stat_peaks()`, `stat_valleys()`, and `stat_poly_eq()`. Peaks and valleys are local (or global) maxima and minima. These stats return the x and y values at the peaks or valleys plus suitable labels,

and default aesthetics that make easy their use with several different geoms, including `geom_point`, `geom_text`, `geom_label`, `geom_vline`, `geom_hline` and `geom_rug`, and also with geoms defined by package 'ggrepel'. Some examples follow.

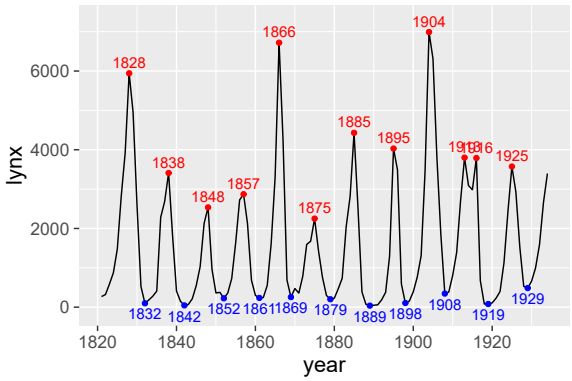
1.2.3 Peaks and valleys

There are many cases, for example in physics and chemistry, but also when plotting time-series data when we need to automatically locate and label local maxima (peaks) or local minima (valleys) in curves. The statistics presented here are useful only for dense data as they do not fit a peak function but instead simply search for the local maxima or minima in the observed data. However, they allow flexible generation of labels on both x and y peak or valley coordinates.

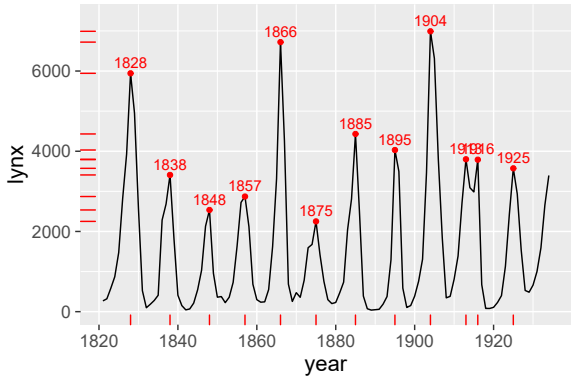
We use as example the same time series as above. In the next several examples we demonstrate some of this flexibility.

```
lynx.df <- data.frame(year = as.numeric(time(lynx)), lynx = as.matrix(lynx))
```

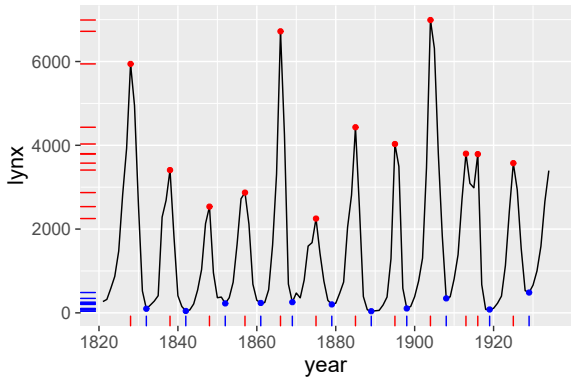
```
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_peaks(colour = "red") +
  stat_peaks(geom = "text", colour = "red",
             vjust = -0.5, x.label.fmt = "%4.0f") +
  stat_valleys(colour = "blue") +
  stat_valleys(geom = "text", colour = "blue",
               vjust = 1.5, x.label.fmt = "%4.0f") +
  ylim(-100, 7300)
```




```
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_peaks(colour = "red") +
  stat_peaks(geom = "rug", colour = "red") +
  stat_peaks(geom = "text", colour = "red",
             vjust = -0.5, x.label.fmt = "%4.0f") +
  ylim(NA, 7300)
```

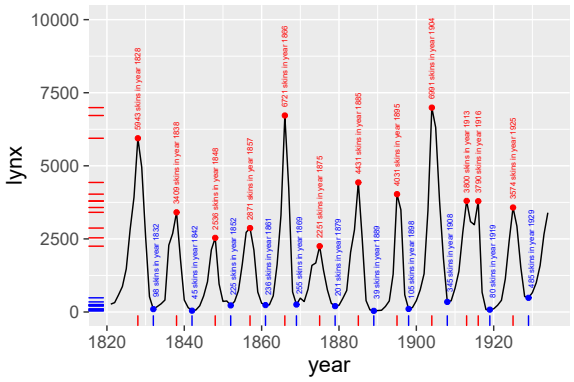



```
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_peaks(colour = "red") +
  stat_peaks(geom = "rug", colour = "red") +
  stat_valleys(colour = "blue") +
  stat_valleys(geom = "rug", colour = "blue")
```



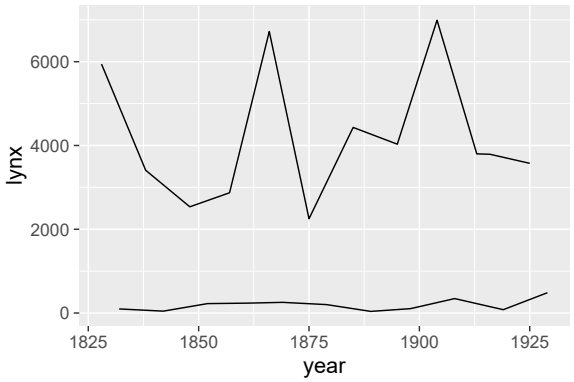

```
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_peaks(colour = "red") +
  stat_peaks(geom = "rug", colour = "red") +
  stat_peaks(geom = "text", colour = "red",
             hjust = -0.1, label.fmt = "%4.0f",
             angle = 90, size = rel(2),
             aes(label = paste(..y.label..,
                               "skins in year", ..x.label..))) +
  stat_valleys(colour = "blue") +
  stat_valleys(geom = "rug", colour = "blue") +
  stat_valleys(geom = "text", colour = "blue",
             hjust = -0.1, vjust = 1, label.fmt = "%4.0f",
             angle = 90, size = rel(2),
```

```
            aes(label = paste(..y.label..,
                              "skins in year", ..x.label..))) +
  ylim(NA, 10000)
```



Of course, if one finds use for it, the peaks and/or valleys can be plotted on their own. Here we plot an "envelope" using `geom_line()`.

```
ggplot(lynx.df, aes(year, lynx)) +
  stat_peaks(geom = "line") + stat_valleys(geom = "line")
```



1.2.4 Equations as labels in plots

How to add a label with a polynomial equation including coefficient estimates from a model fit seems to be a frequently asked question in Stackoverflow. The parameter estimates are extracted automatically from a fit object corresponding to each *group* or panel in a plot and other aesthetics for the group respected. An aesthetic is provided for this, and only this. Such a statistics needs to be used

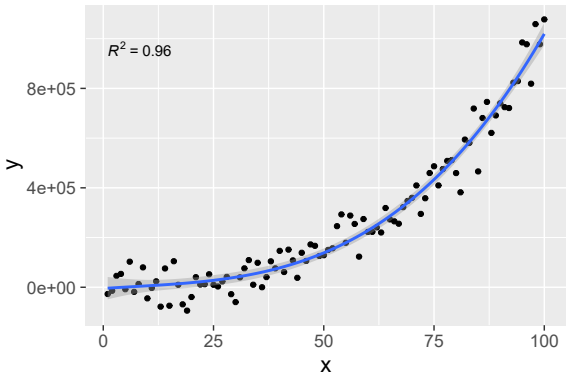
together with another geom or stat like geom smooth to add the fitted line. A different approach, discussed in Stackoverflow, is to write a statistics that does both the plotting of the polynomial and adds the equation label. Package 'ggpmisc' defines `stat_poly_eq()` using the first approach which follows the 'rule' of using one function in the code for a single action. In this case there is a drawback that the users is responsible for ensuring that the model used for the label and the label are the same, and in addition that the same model is fitted twice to the data.

We first generate some artificial data.

```
set.seed(4321)
# generate artificial data
x <- 1:100
y <- (x + x^2 + x^3) +
  rnorm(length(x), mean = 0, sd = mean(x^3) / 4)
my.data <- data.frame(x, y,
                      group = c("A", "B"),
                      y2 = y * c(0.5,2))
```

First an example using default arguments.

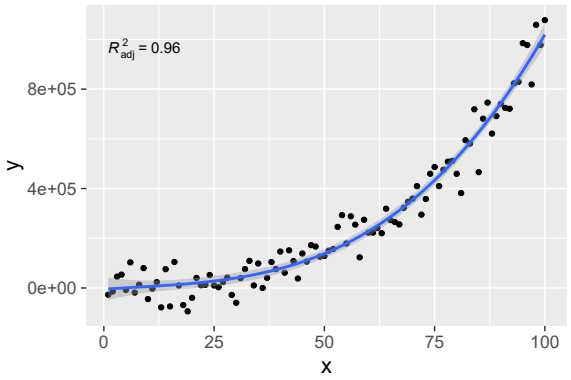
```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(formula = formula, parse = TRUE)
```



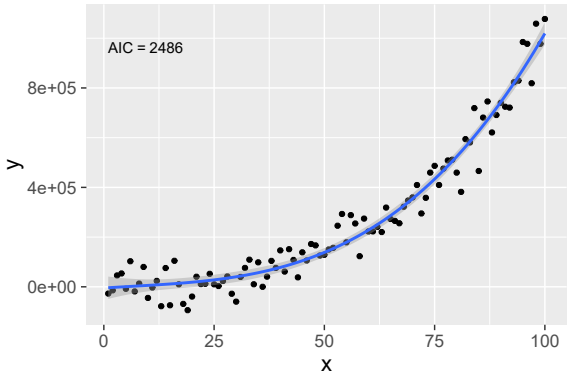
`stat_poly_eq()` makes available five different labels in the returned data frame. R^2, $R_mathrmadj^2$, AIC, BIC and the polynomial equation. R^2 is used by default, but `aes()` can be used to select a different one.

```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
```

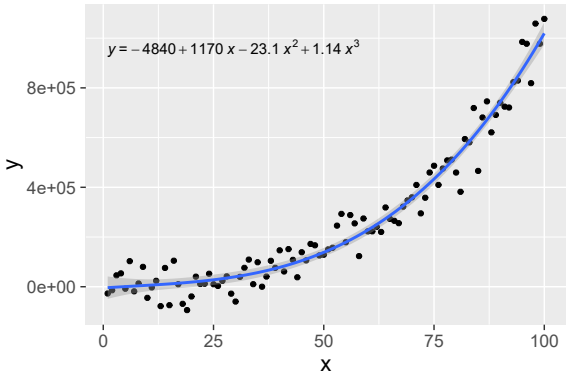
```
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..adj.rr.label..),
               formula = formula, parse = TRUE)
```




```
formula <- y ~ poly(x, 3, raw = TRUE)
  ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..AIC.label..),
               formula = formula, parse = TRUE)
```

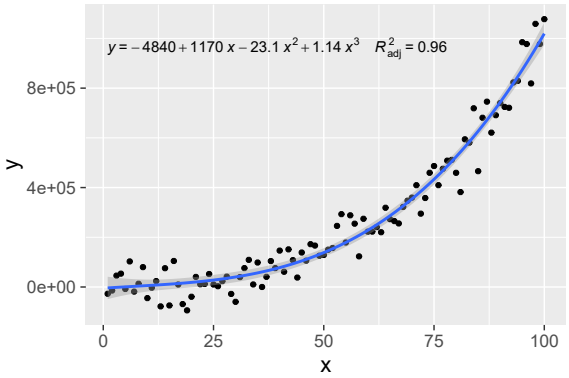



```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..),
               formula = formula, parse = TRUE)
```



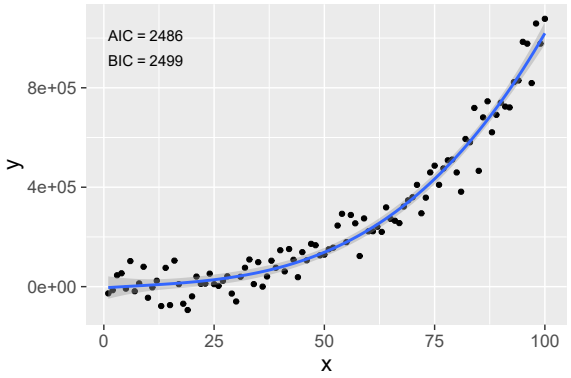
Within `aes()` it is possible to *compute* new labels based on those returned plus "arbitrary" text. The supplied labels are meant to be *parsed* into R expressions, so any text added should be valid for a string that will be parsed.

```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = paste(..eq.label..,
                                 ..adj.rr.label..,
                                 sep = "~~~~")),
               formula = formula, parse = TRUE)
```



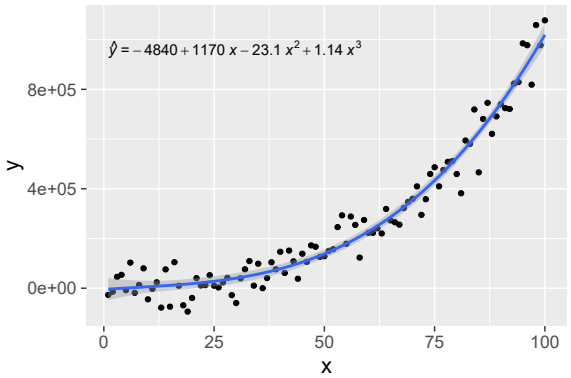

```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = paste("atop(", ..AIC.label.., ",",
                                 ..BIC.label.., ")",
```

```
                                    sep = "")),
               formula = formula, parse = TRUE)
```



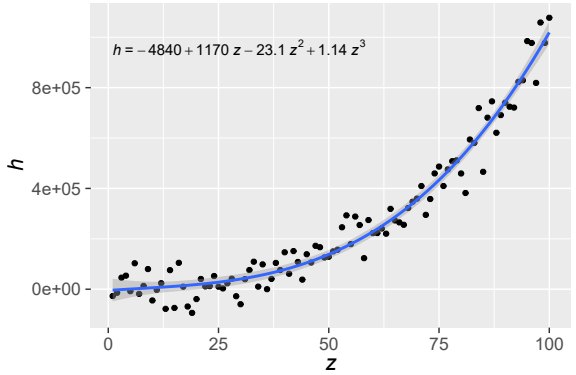
Two examples of removing or changing the *lhs* and/or the *rhs* of the equation. (Be aware that the equals sign must be always enclosed in backticks in a string that will be parsed.)

```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..),
               eq.with.lhs = "italic(hat(y))~`=`~",
formula = formula, parse = TRUE)
```



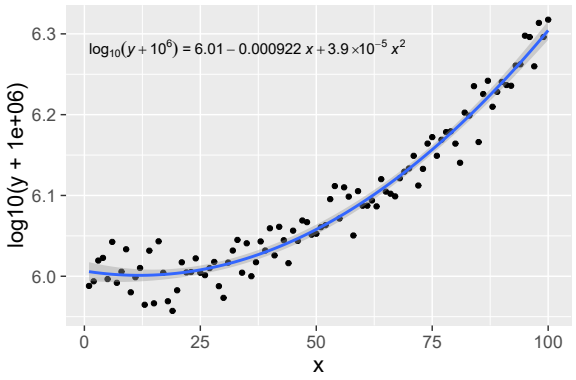

```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
```

```
  geom_smooth(method = "lm", formula = formula) +
  labs(x = expression(italic(z)), y = expression(italic(h)) ) +
  stat_poly_eq(aes(label = ..eq.label..),
               eq.with.lhs = "italic(h)~`=`~",
               eq.x.rhs = "~italic(z)",
               formula = formula, parse = TRUE)
```



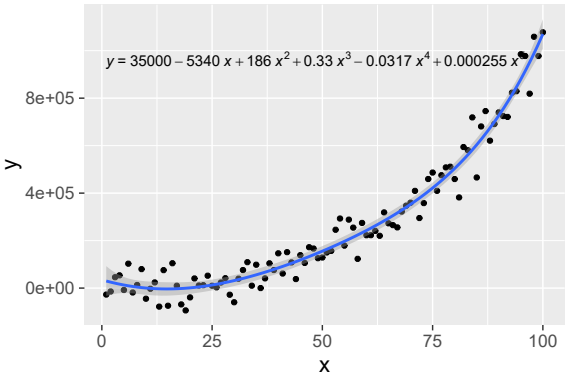
As any valid R expression can be used, Greek letters are also supported, as well as the inclusion in the label of variable transformations used in the model formula.

```
formula <- y ~ poly(x, 2, raw = TRUE)
ggplot(my.data, aes(x, log10(y + 1e6))) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..),
               eq.with.lhs = "plain(log)[10](italic(y)+10^6)~`=`~",
               formula = formula, parse = TRUE)
```



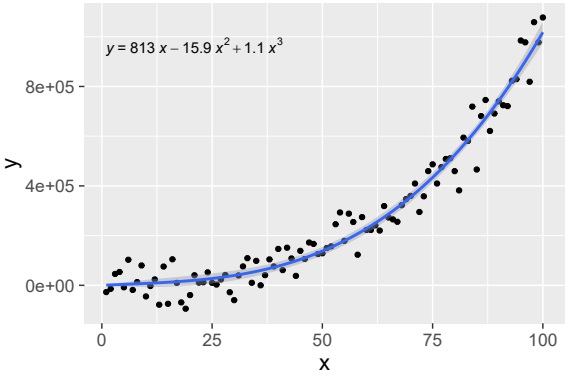
Example of a polynomial of fifth order.

```
formula <- y ~ poly(x, 5, raw = TRUE)
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..),
               formula = formula, parse = TRUE)
```



Intercept forced to zero—line through the origin.

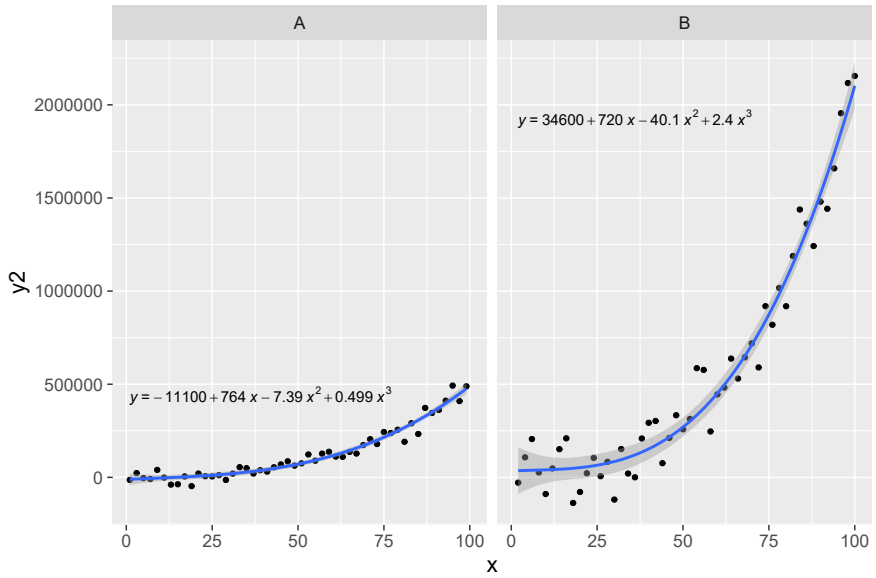
```
formula <- y ~ x + I(x^2) + I(x^3) - 1
ggplot(my.data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..),
               formula = formula, parse = TRUE)
```



We give some additional examples to demonstrate how other components of the `ggplot` object affect the behaviour of this statistic.

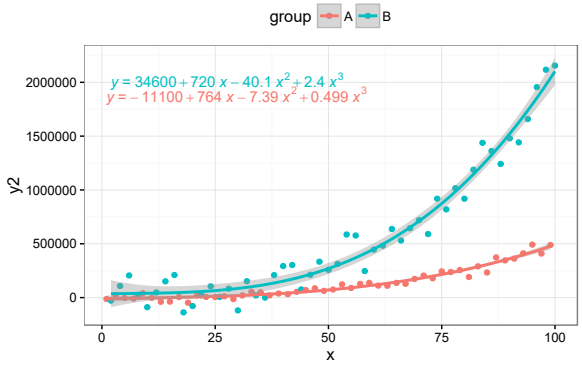
Facets work as expected either with fixed or free scales. Although bellow we had to adjust the size of the font used for the equation.

```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y2)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..), # size = rel(2.8),
               formula = formula, parse = TRUE) +
  facet_wrap(~group)
```



Grouping, in this example using colour aesthetic also works as expected, but we need to use vjust to prevent the two equation labels to overlap.

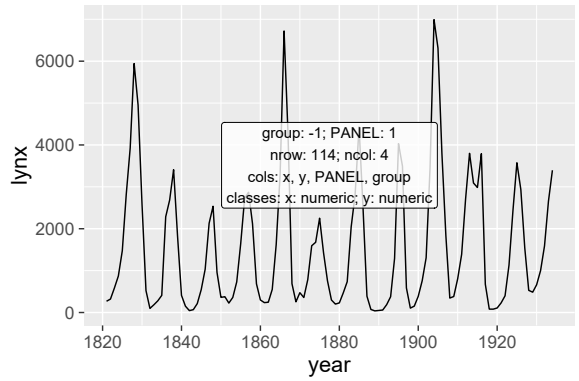
```
formula <- y ~ poly(x, 3, raw = TRUE)
ggplot(my.data, aes(x, y2, colour = group)) +
  geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label = ..eq.label..), vjust = c(-8, 0),
               formula = formula, parse = TRUE) +
  theme_bw() +
  theme(legend.position = "top")
```



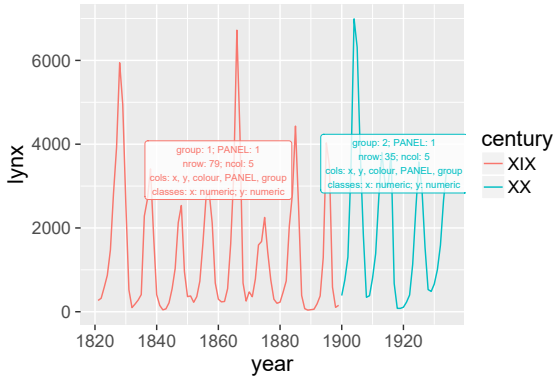
1.2.5 Learning and/or debugging

A very simple stat named `stat_debug()` can save the work of adding print statements to the code of stats to get information about what data is being passed to the `compute_group()` function. Because the code of this function is stored in a `ggproto` object, at the moment it is impossible to directly set breakpoints in it. This `stat_debug()` may also help users diagnose problems with the mapping of aesthetics in their code or just get a better idea of how the internals of 'ggplot2' work.

```
ggplot(lynx.df, aes(year, lynx)) + geom_line() +
  stat_debug_group(alpha = 0.8)
```




```
lynx.df$century <- ifelse(lynx.df$year >= 1900, "XX", "XIX")
ggplot(lynx.df, aes(year, lynx, color = century)) +
  geom_line() +
  stat_debug_group(alpha = 0.8, size = rel(2.5))
```



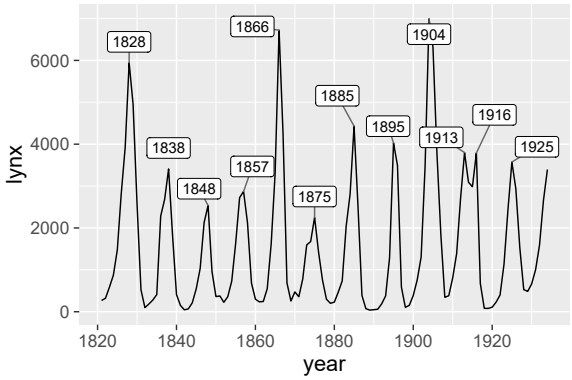
1.3 'ggrepel'

Package 'ggrepel' is under development by Kamil Slowikowski. It does a single thing, relocates text labels so that they do not overlap. This is achieved through two geometries that work similarly to those provided by 'ggplot2' except for the relocation. This is incredibly useful both when labeling peaks and valleys and when labeling points in scatter-plots. This is a significant problem in bioinformatics plots and in maps.

1.3.1 New geoms

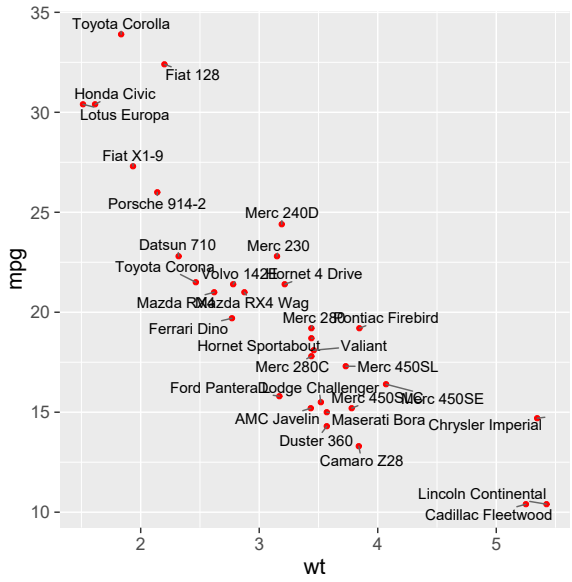
Package 'ggrepel' provides two new geoms: `geom_text_repel` and `geom_label_repel`. They are used similarly to `geom_text` and `geom_label` but the text or labels "repel" each other so that they rarely overlap unless the plot is very crowded.

```
ggplot(lynx.df, aes(year, lynx)) +
  geom_line() +
  stat_peaks(geom = "label_repel", nudge_y = 500)
```



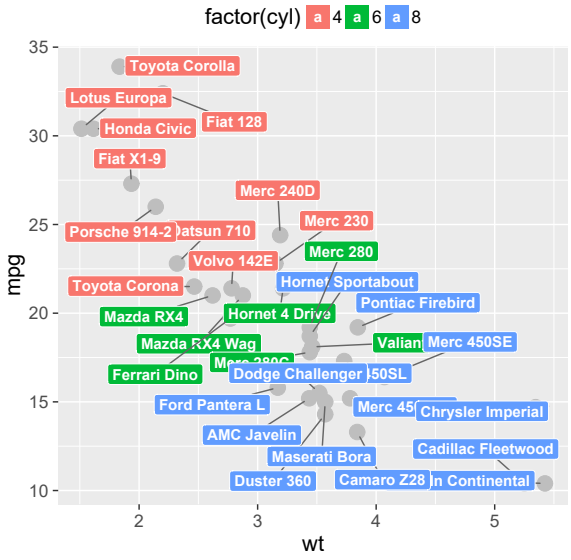
For the time being I reproduce here a couple of examples from the package vignette.

```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(color = 'red') +
  geom_text_repel(aes(label = rownames(mtcars)))
```

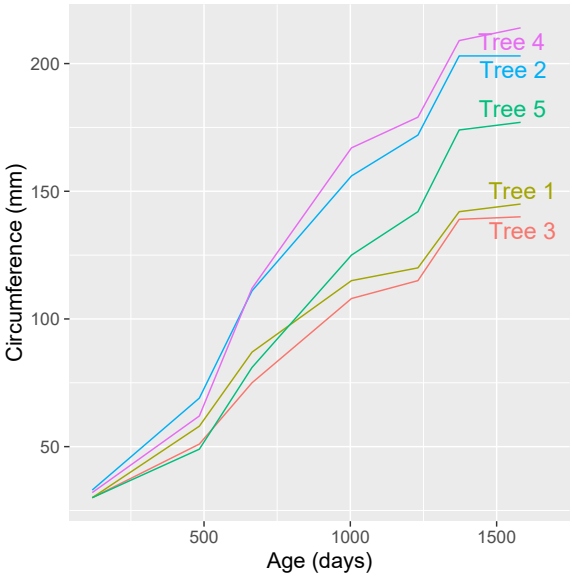



```
set.seed(42)
ggplot(mtcars) +
  geom_point(aes(wt, mpg), size = 5, color = 'grey') +
  geom_label_repel(
```

```
  aes(wt, mpg, fill = factor(cyl), label = rownames(mtcars)),
      fontface = 'bold', color = 'white',
      box.padding = unit(0.25, "lines"),
      point.padding = unit(0.5, "lines")) +
  theme(legend.position = "top")
```




```
set.seed(42)
ggplot(Orange, aes(age, circumference, color = Tree)) +
  geom_line() +
  coord_cartesian(xlim = c(min(Orange$age), max(Orange$age) + 90)) +
  geom_text_repel(data = subset(Orange, age == max(age)),
                  aes(label = paste("Tree", Tree)),
                  size = 6,
                  nudge_x = 45,
                  segment.color = NA) +
  theme(legend.position = "none") +
  labs(x = "Age (days)", y = "Circumference (mm)")
```



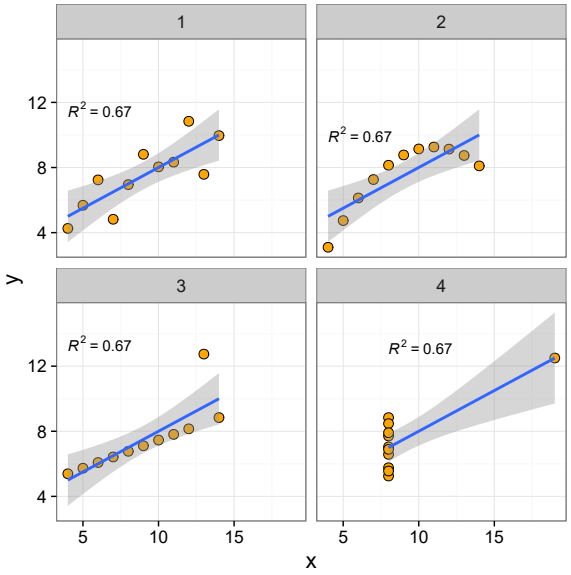
1.4 Examples

1.4.1 Anscombe's example revisited

To make the chapter self contained we repeat the code from chapter **??**.

```
# we rearrange the data
my.mat <- matrix(as.matrix(anscombe), ncol=2)
my.anscombe <- data.frame(x = my.mat[ , 1],
                          y = my.mat[ , 2],
                          case=factor(rep(1:4, rep(11,4))))
```

```
ggplot(my.anscombe, aes(x,y)) +
  geom_point(shape=21, fill="orange", size=3) +
  geom_smooth(method="lm") +
  stat_poly_eq(formula = y ~ x, vjust = -1, parse = TRUE) +
  facet_wrap(~case, ncol=2) +
  theme_bw(16)
```




```
try(detach(package:ggpmisc))
try(detach(package:ggrepel))
try(detach(package:ggplot2))
```

2 Further reading about R

2.1 Introductory texts

2.2 Texts on specific aspects

2.3 Advanced texts

Bibliography

Chang, W. (2013). *R Graphics Cookbook*. 1-2. Sebastopol: O'Reilly Media, p. 413. ISBN: 9781449316952. URL: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf.

Dalgaard, P. (2008). *Introductory Statistics with R*. Springer, p. 380. ISBN: 0387790543.

Everitt, B. and T. Hothorn (2011). *An Introduction to Applied Multivariate Analysis with R*. Springer, p. 288. ISBN: 1441996494. URL: http://www.amazon.co.uk/Introduction-Applied-Multivariate-Analysis-Use/dp/1441996494.

Faraway, J. J. (2004). *Linear Models with R*. Boca Raton, FL: Chapman & Hall/CRC, p. 240. URL: http://www.maths.bath.ac.uk/~jjf23/LMR/.

Faraway, J. J. (2006). *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. Chapman & Hall/CRC Taylor & Francis Group, p. 345. ISBN: 158488424X.

Fox, J. (2002). *An {R} and {S-Plus} Companion to Applied Regression*. Thousand Oaks, CA, USA: Sage Publications. URL: http://socserv.socsci.mcmaster.ca/jfox/Books/Companion/index.html.

Fox, J. and H. S. Weisberg (2010). *An R Companion to Applied Regression*. SAGE Publications, Inc, p. 472. ISBN: 141297514X. URL: http://www.amazon.com/An-R-Companion-Applied-Regression/dp/141297514X.

Ihaka, R. and R. Gentleman (1996). 'R: A Language for Data Analysis and Graphics'. In: *J. Comput. Graph. Stat.* 5, pp. 299–314.

Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press, p. 400. ISBN: 1593273843. URL: http://www.amazon.com/The-Art-Programming-Statistical-Software/dp/1593273843.

Murrell, P. (2011). *R Graphics, Second Edition (Chapman & Hall/CRC The R Series)*. CRC Press, p. 546. ISBN: 1439831769. URL: http://www.amazon.com/Graphics-Second-Edition-Chapman-Series/dp/1439831769.

Pinheiro, J. C. and D. M. Bates (2000). *Mixed-Effects Models in S and S-Plus*. New York: Springer.

Teetor, P. (2011). *R Cookbook*. 1st ed. Sebastopol: O'Reilly Media, p. 436. ISBN: 9780596809157.

Wickham, H. (2014). *Advanced R*. Chapman & Hall/CRC The R Series. CRC Press. ISBN: 9781466586970. URL: https://books.google.fi/books?id=G5PNBQAAQBAJ.

Wickham, H. (2015). *R Packages*. O'Reilly Media. ISBN: 9781491910542. URL: https://books.google.fi/books?id=eqOxBwAAQBAJ.

Xie, Y. (2013). *Dynamic Documents with R and knitr (Chapman & Hall/CRC The R Series)*. Chapman and Hall/CRC, p. 216. ISBN: 1482203537. URL: http://www.amazon.com/Dynamic-Documents-knitr-Chapman-Series/dp/1482203537.

Zuur, A. F., E. N. Ieno and E. Meesters (2009). *A Beginner's Guide to R*. 1st ed. Springer, p. 236. ISBN: 0387938362. URL: http://www.amazon.com/Beginners-Guide-Use-Alain-Zuur/dp/0387938362.