

# Lab07

● Graded

Student

Sangwon Ji

Total Points

4 / 4 pts

Question 1

1a

1 / 1 pt

✓ - 0 pts Correct

Question 2

1b

1 / 1 pt

✓ - 0 pts Correct

Question 3

1c

1 / 1 pt

✓ - 0 pts Correct

Question 4

2

1 / 1 pt

✓ - 0 pts Correct

No questions assigned to the following page.

# Lab 07

In this lab you will:

- Explore some details for making plots
- Work with polynomial models

## Plots

We will continue working with the Google Trend dataset for R and Python. Recall the dataset we obtained contains the following variables:

- **week**: beginning date of the week (recent 5 years)
- **python**: trend of the search term **Data science Python**
- **r**: trend of the search term **Data science r**

Read the data.

```
data_science <- read.csv("data_science.csv")
# convert string to date object
data_science$week <- as.Date(data_science$week, "%Y-%m-%d")
# create a numeric column representing the time
data_science$time <- as.numeric(data_science$week)
data_science$time <- data_science$time - data_science$time[1] + 1
```

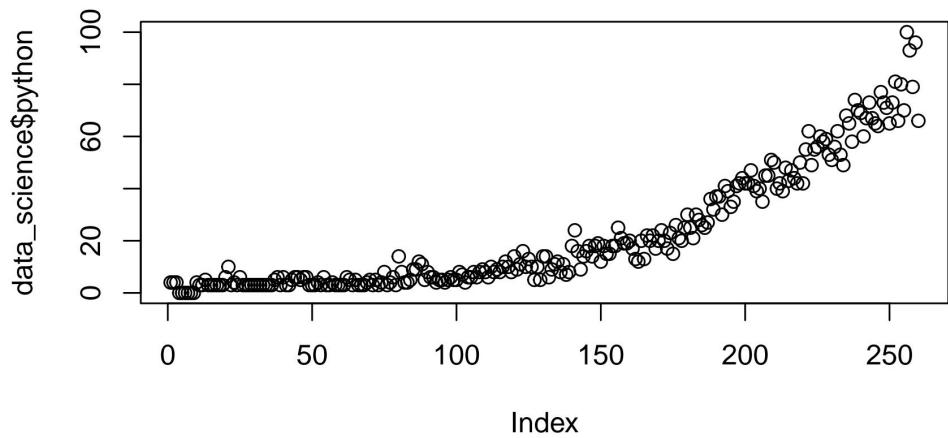
### Function plot

The function **plot** is the most standard plotting function in R. When taking only one vector, it plots the vector against the index vector.

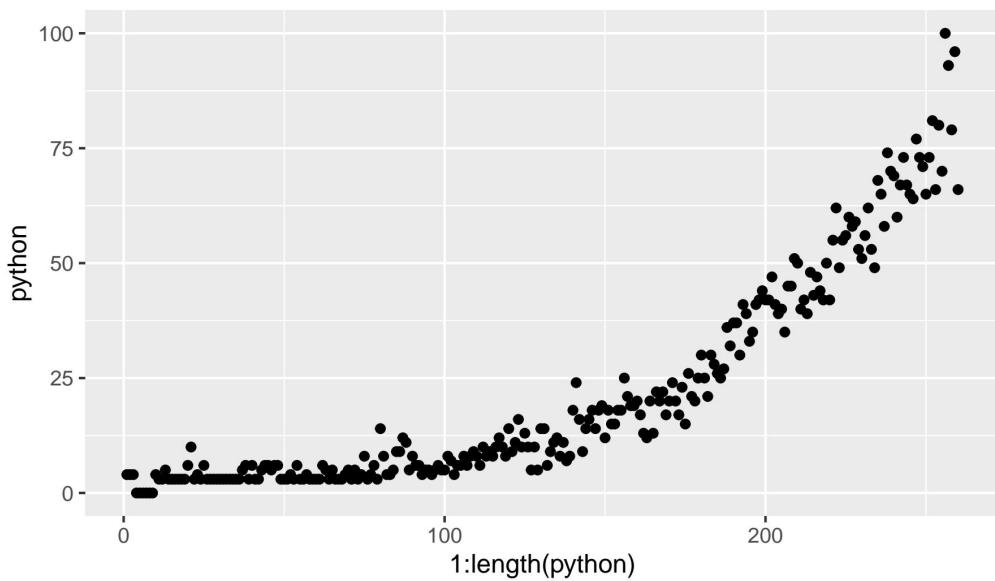
```
plot(data_science$python)
```

```
library(ggplot2)
```

No questions assigned to the following page.



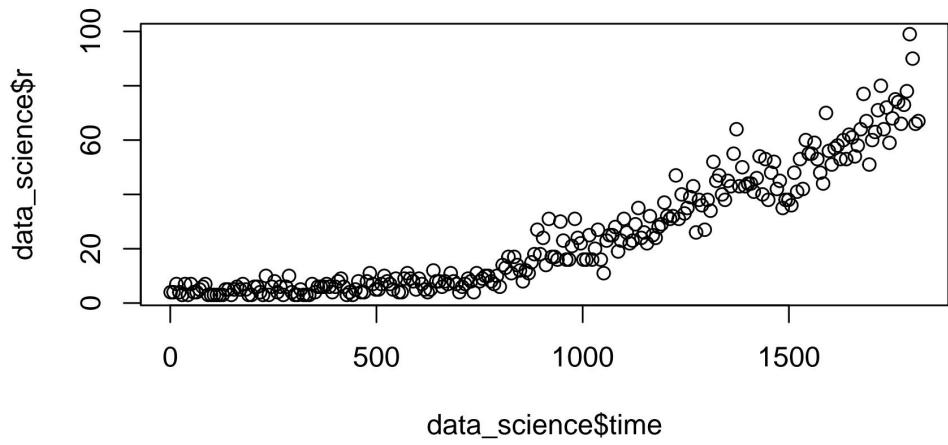
```
#tidyverse  
ggplot(data_science)+geom_point(aes(x=1:length(python), y=python))
```



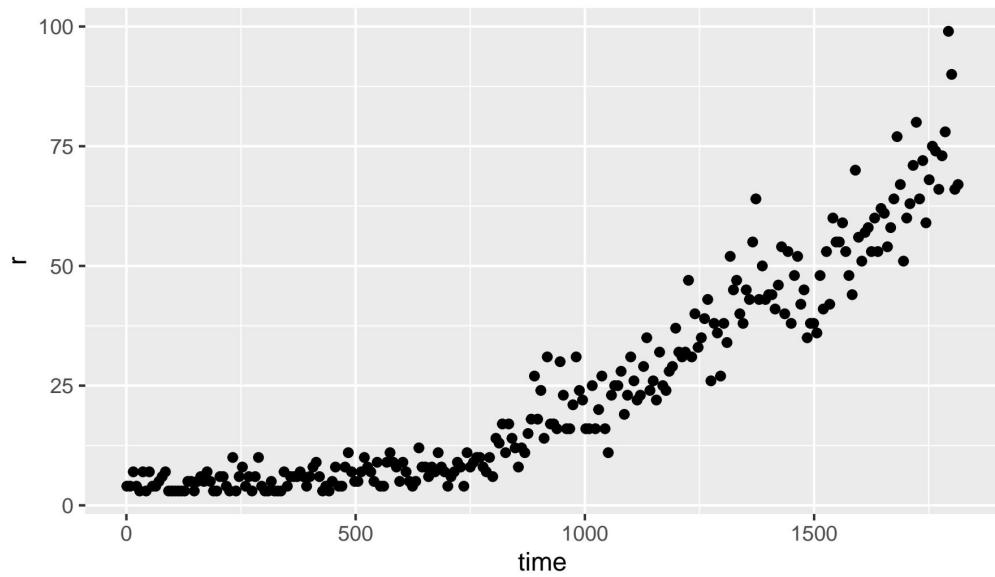
When taking two vectors, it plots the scatter plot.

```
plot(data_science$time, data_science$r)
```

No questions assigned to the following page.



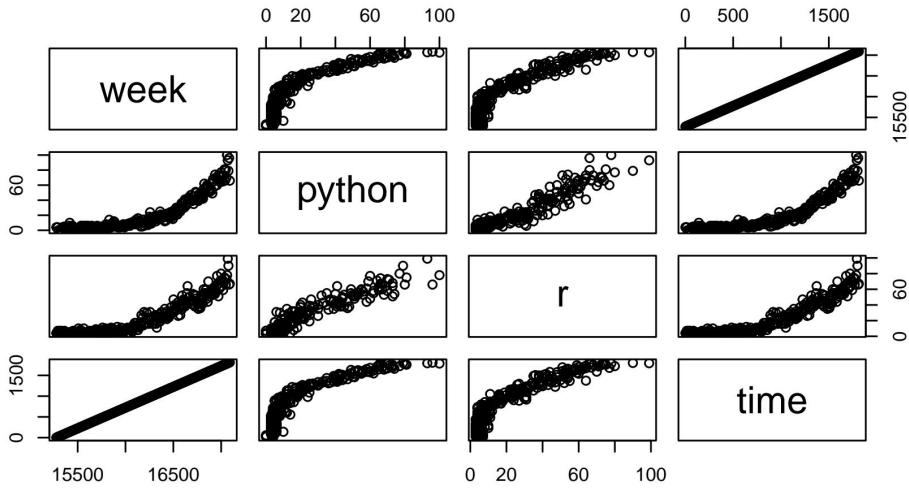
```
#tidyverse
ggplot(data_science)+geom_point(aes(x=time, y=r))
```



The `plot` function can also accept a data frame as the parameter. And it plots all variables against each other.

```
plot(data_science)
```

No questions assigned to the following page.



### Low-level graphics functions

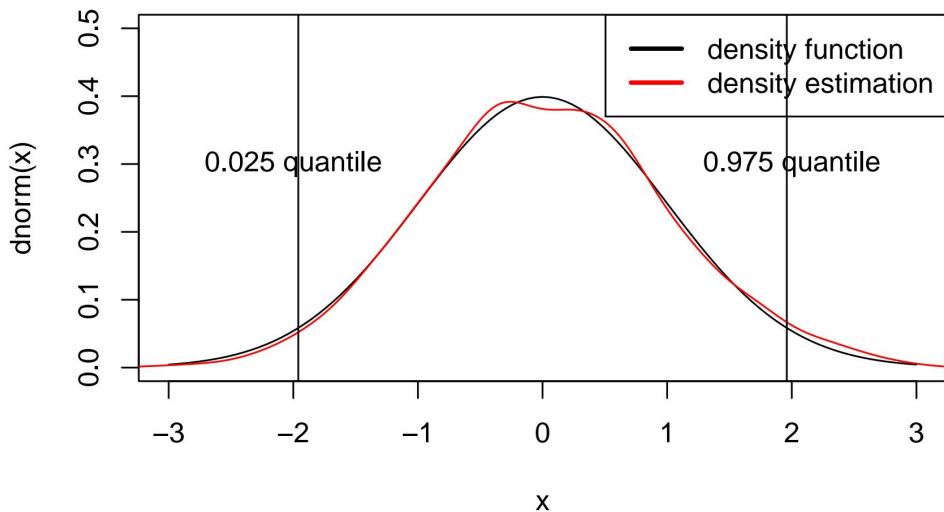
The `plot` function introduced above is a high-level plot function, which produces complete plots. There are also low-level functions that add further outputs to an existing plot. You have already seen some high-level functions such as `hist`, `boxplot` and `curve`. `lines` is a low-level function which can not be called directly.

Commonly used low-level functions includes:

- `points(x, y)`: Adds points to the current plot. `x` and `y` are vectors of coordinates.
- `lines(x, y)`: Add line to the current plot by joining the points with line segments.
- `text(x, y, labels, ...)`: Add text to a plot at points given by `x`, `y`.
- `abline(a, b)`: Adds a line of slope `b` and intercept `a` to the current plot.
- `abline(h=y)`: Adds a horizontal line.
- `abline(v=x)`: Adds a vertical line.
- `legend(x, y, legend, ..., lty = c())`: Add legends to plots

```
curve(dnorm, xlim = c(-3, 3), ylim = c(0, 0.5)) # create a plot for the density of normal distribution
lines(density(rnorm(1000)), col = "red") # add a line (kernel density estimation) to the plot created
abline(v=qnorm(0.025)) # add a vertical line (0.025 quantile of standard normal distribution)
abline(v=qnorm(0.975)) # add a vertical line (0.975 quantile of standard normal distribution)
text(-2, 0.3, "0.025 quantile") # add text
text(2, 0.3, "0.975 quantile") # add text
legend("topright", c("density function", "density estimation"),
       lwd=c(2.5,2.5), # Line widths in the legend
       col=c("black", "red")) # Add legend
```

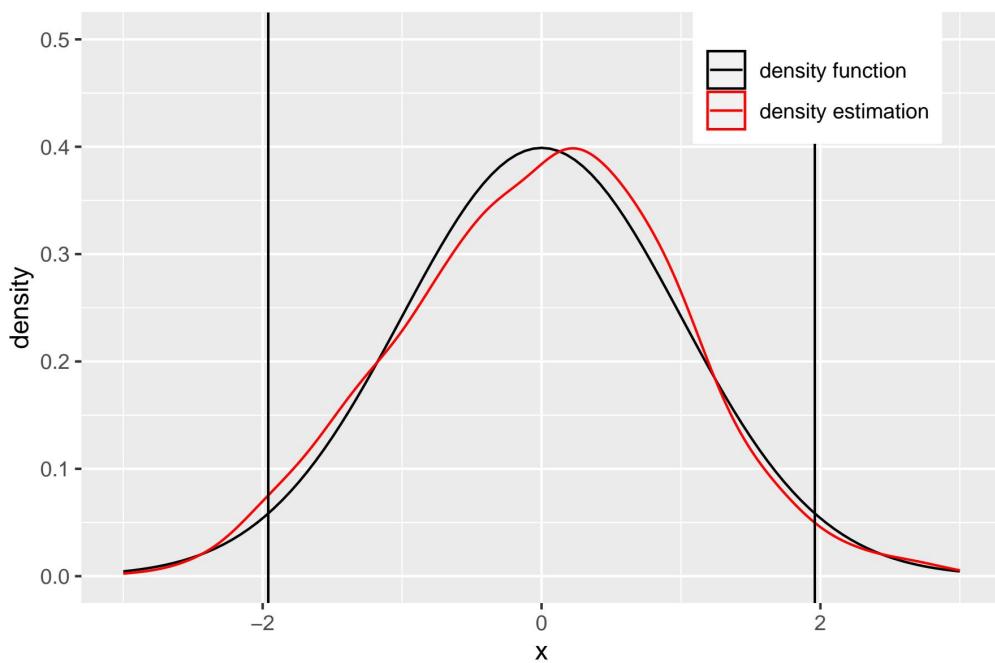
No questions assigned to the following page.



```
#tidyverse
ggplot(data = data.frame(data = rnorm(1000))+
  stat_function(fun = dnorm, mapping=aes(color = 'density function'))+
  geom_density(aes(x=data, color = 'density estimation'))+
  geom_vline(xintercept=qnorm(0.025))+ 
  geom_vline(xintercept=qnorm(0.975))+ 
  scale_colour_manual("", 
                      breaks = c("density function", "density estimation"),
                      values = c("black", "red"))+
  xlab("x")+
  xlim(-3,3)+
  ylim(0,0.5)+
  theme(legend.position = c(0.8, 0.9)))

## Warning: Removed 3 rows containing non-finite values (`stat_density()`).
```

No questions assigned to the following page.



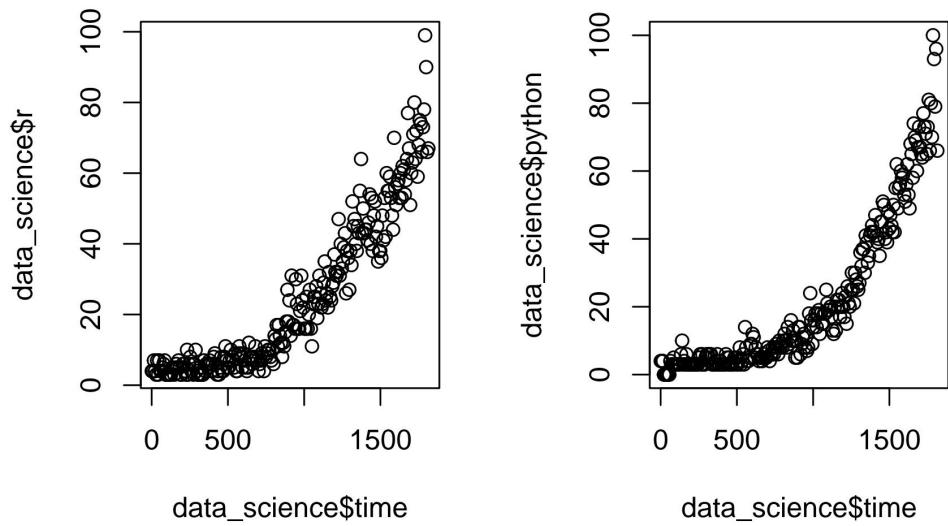
```
par()
```

The `par()` function is used to access and modify the list of graphics parameters. For example, to put several plots in the same window, use `par(mfrow = c(a, b))`. `a` is the number of rows and `b` is the number of columns. This command will allow you to plot `a*b` plots in one window.

```
library(tidyr)
```

```
par(mfrow=c(1, 2))
plot(data_science$time, data_science$r)
plot(data_science$time, data_science$python)
```

No questions assigned to the following page.

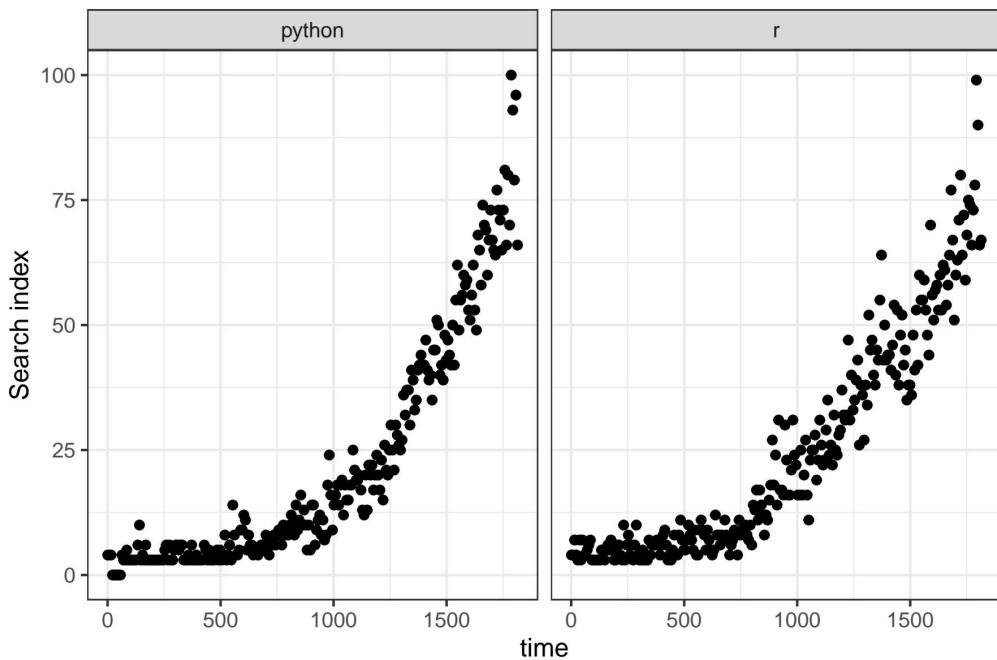


```
#tidyverse

data_wide = pivot_longer(data_science,
  cols = c(r, python),
  names_to = "Language",
  values_to = "Search index")

ggplot(data_wide, aes(x = time, y = `Search index`)) +
  geom_point() + facet_wrap(. ~ Language) +
  theme_bw()
```

No questions assigned to the following page.



### patchwork

There is a useful package to combine separate ggplots into the same graphic, “patchwork”.

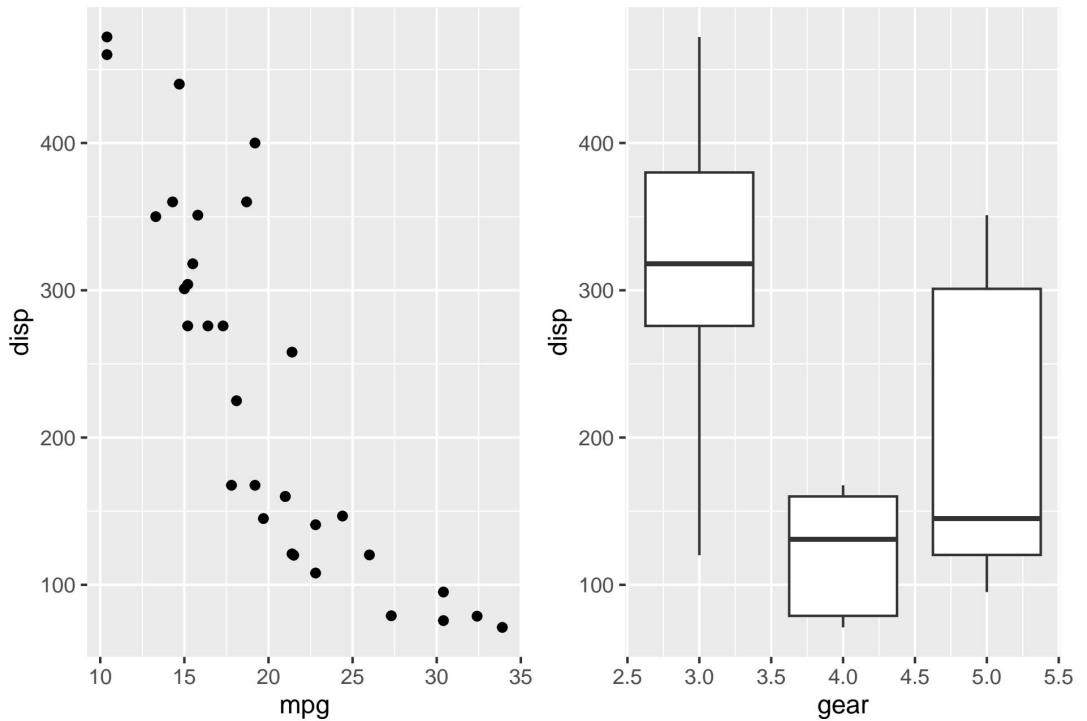
```
# Uncomment the following lines to install the packages
# install.packages("devtools")
# devtools::install_github("thomasp85/patchwork")

library(ggplot2)
library(patchwork)

p1 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))

p1 + p2
```

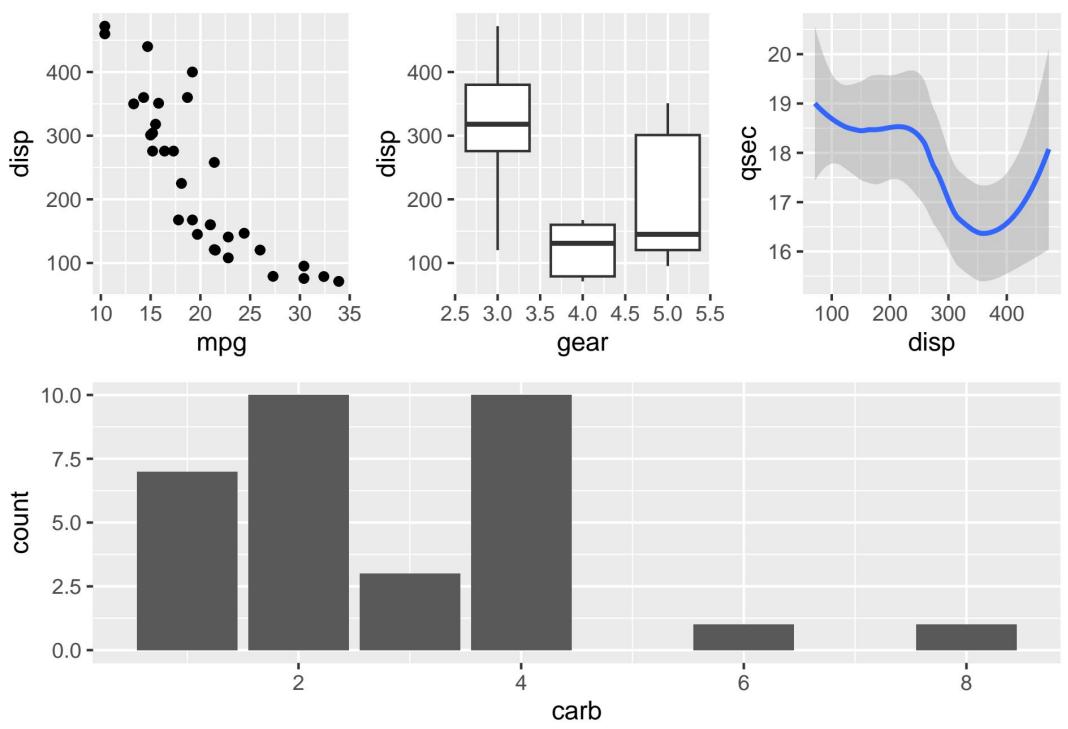
No questions assigned to the following page.



```
p3 <- ggplot(mtcars) + geom_smooth(aes(disp, qsec), formula = 'y~x' ,method = 'loess')
p4 <- ggplot(mtcars) + geom_bar(aes(carb))
```

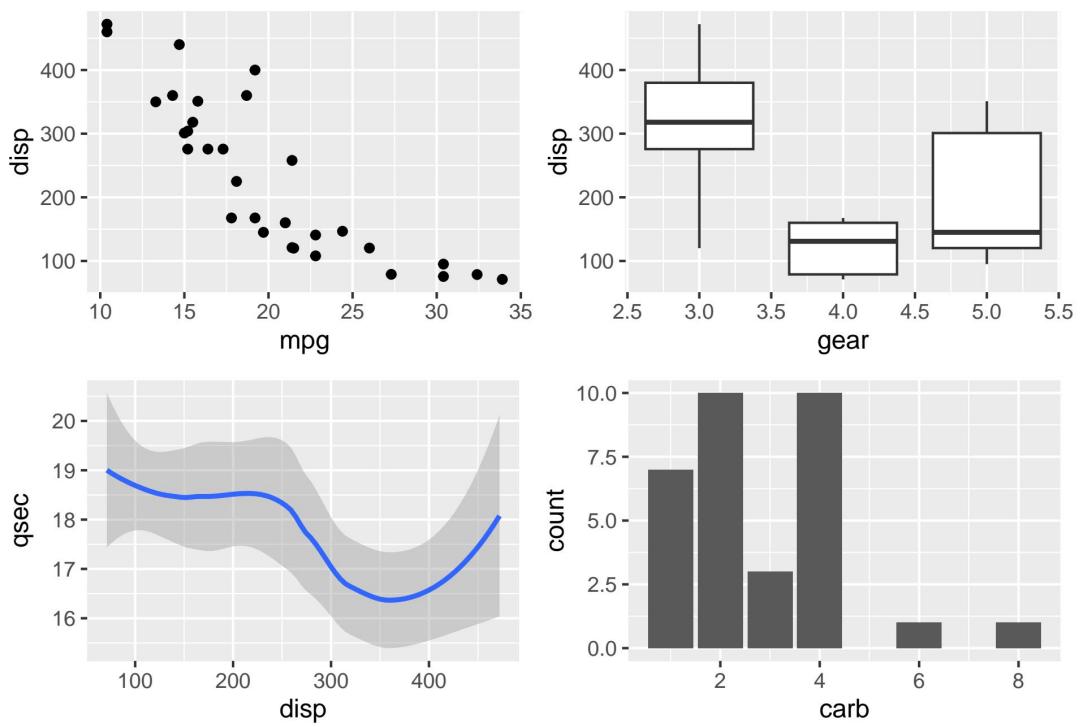
```
(p1 | p2 | p3) /
  p4
```

No questions assigned to the following page.



$(p_1 \mid p_2) / (p_3 \mid p_4)$

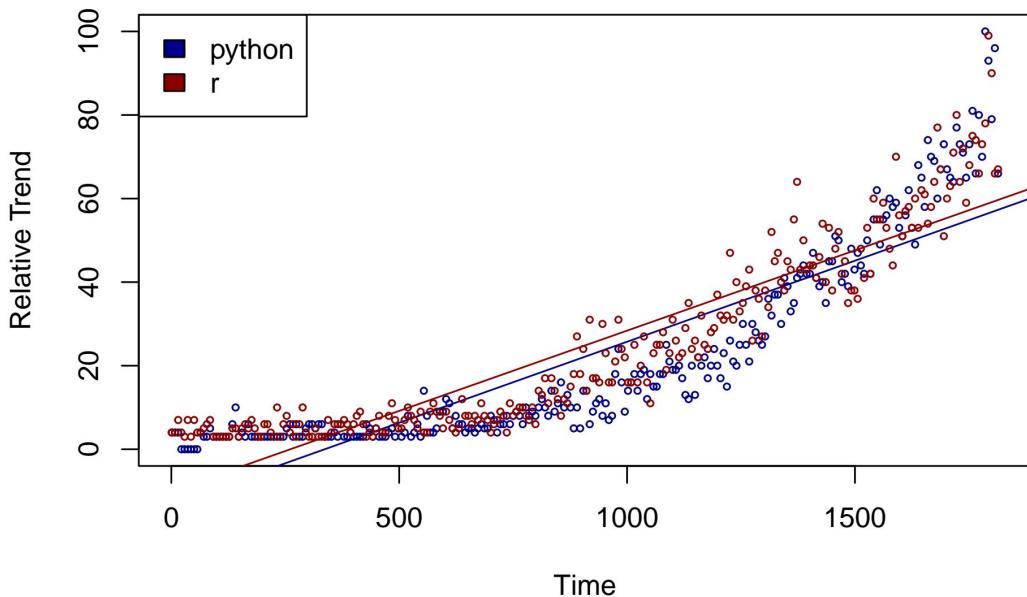
No questions assigned to the following page.



Other functions or packages for arranging ggplots: `gridExtra::grid.arrange()` and `cowplot::plot_grid`.

Question assigned to the following page: [1](#)

## Google trend of data science languages



## Polynomial models

### Exercise 1

As you may discover, the linear model is not quite good for your dataset. In this exercise, you will fit a cubic curve to the data (use time to predict r and python).

- (a) Fit the cubic model.

```
# Insert your code here and save your fitted model as
# `python.poly` and `r.poly`
python.poly <- lm(python ~ time + I(time^2) + I(time^3), data_science)
summary(python.poly)
```

```
##
## Call:
## lm(formula = python ~ time + I(time^2) + I(time^3), data = data_science)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.8268 -1.7714 -0.2982  2.1472 19.3062
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.591e+00 1.039e+00  2.495  0.0132 *
## time        6.970e-03 4.963e-03  1.404  0.1614
## I(time^2) -1.553e-05 6.359e-06 -2.443  0.0153 *
```

Questions assigned to the following page: [2](#) and [3](#)

```

## I(time^3)    2.022e-08  2.303e-09   8.781 2.37e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.23 on 256 degrees of freedom
## Multiple R-squared:  0.967, Adjusted R-squared:  0.9666
## F-statistic:  2498 on 3 and 256 DF,  p-value: < 2.2e-16
# r.poly <-
# summary(r.poly)

```

- (b) In the fitting for r and python search index, which of the following term is significant (not equal to zero)?

```

# Uncomment the line of your answer for this question: (r)
# intercept.r.sig <- TRUE
# time.r.sig <- TRUE
# time.r.square.sig <- TRUE
time.r.cubic.sig <- TRUE

# Uncomment the line of your answer for this question: (python)
# intercept.python.sig <- TRUE
time.python.sig <- TRUE
# time.python.square.sig <- TRUE
# time.python.cubic.sig <- TRUE

```

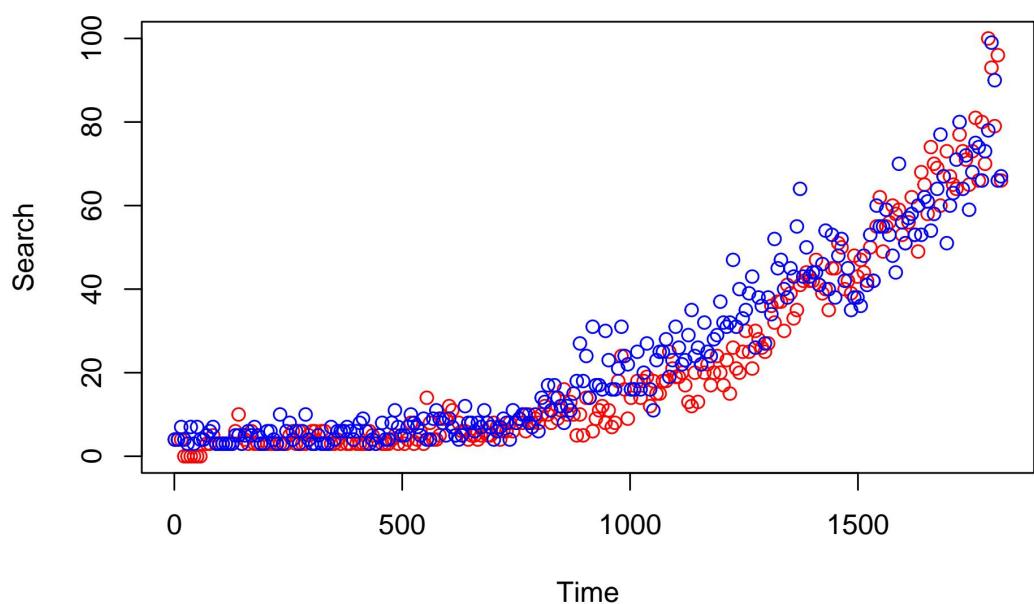
- (c) Plot the scatter plot and the fitted line. Color the groups with red and blue. How will you describe the trend of r search and python search?

```

# Insert your code here for plotting
plot(data_science$time, data_science$python, col= "red", xlab = "Time", ylab = "Search")
points(data_science$time, data_science$r, col="blue")

```

Question assigned to the following page: [3](#)



```
# The trend seems to have a positive linear regression between the two.
```

No questions assigned to the following page.

## Reading - Caveat

Google trend seems so powerful and accessible. However, when analyzing some topics, we would rather not to use it. Why? Consider the following two examples.

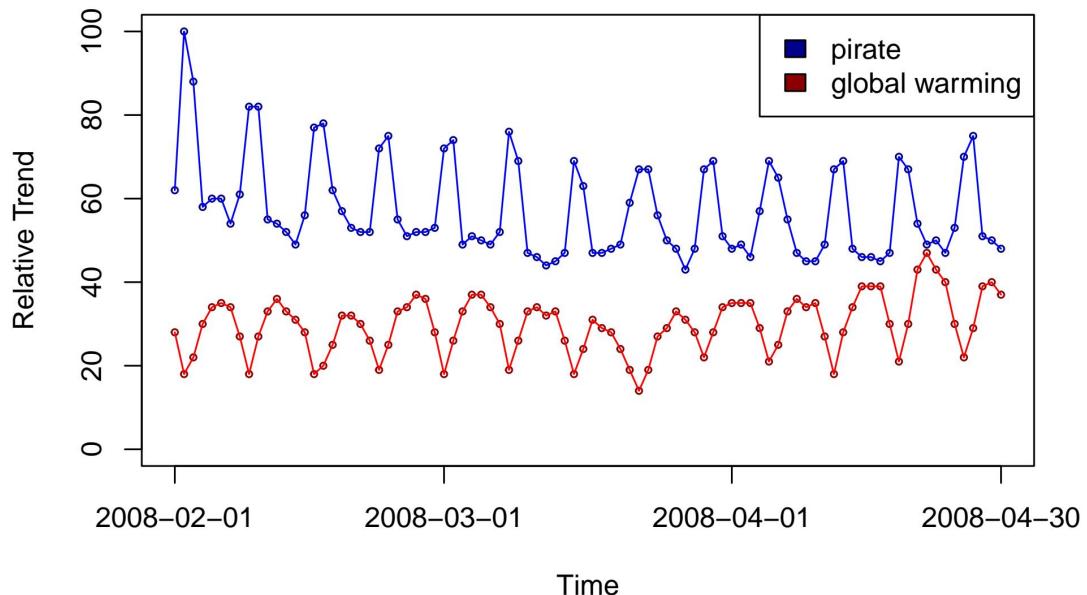
### Global Warming & Pirates

Back to 2008, there is an obvious negative correlation between global warming and pirate searching, which may just be a coincidence. If you fit a linear model, the coefficients are significant.

```
pirate <- read.csv("pirate.csv")
pirate$Day <- as.Date(pirate$Day, "%Y-%m-%d")
data_science$time <- as.numeric(data_science$week)
data_science$time <- as.numeric(data_science$week) - as.numeric(data_science$week)[1]

plot(unclass(pirate$Day), pirate$pirates, cex = 0.5, col = "blue4",
     xlab = "Time", ylab = "Relative Trend", ylim = c(0, 100),
     main = "Google trend of priate and global warming", xaxt='n')
axis(1, at = pirate$Day[c(1, 30, 61, 90)],
     labels = as.character(pirate$Day)[c(1, 30, 61, 90)])
points(pirate$Day, pirate$global.warming, cex = 0.5, col = "red4")
lines(pirate$Day, pirate$pirates, col = "blue")
lines(pirate$Day, pirate$global.warming, col = "red")
legend("topright", legend = c("pirate", "global warming"),
       fill = c("blue4", "red4"))
```

Google trend of piate and global warming



```
pirate.lm <- lm(global.warming ~ pirates, data = pirate)
```

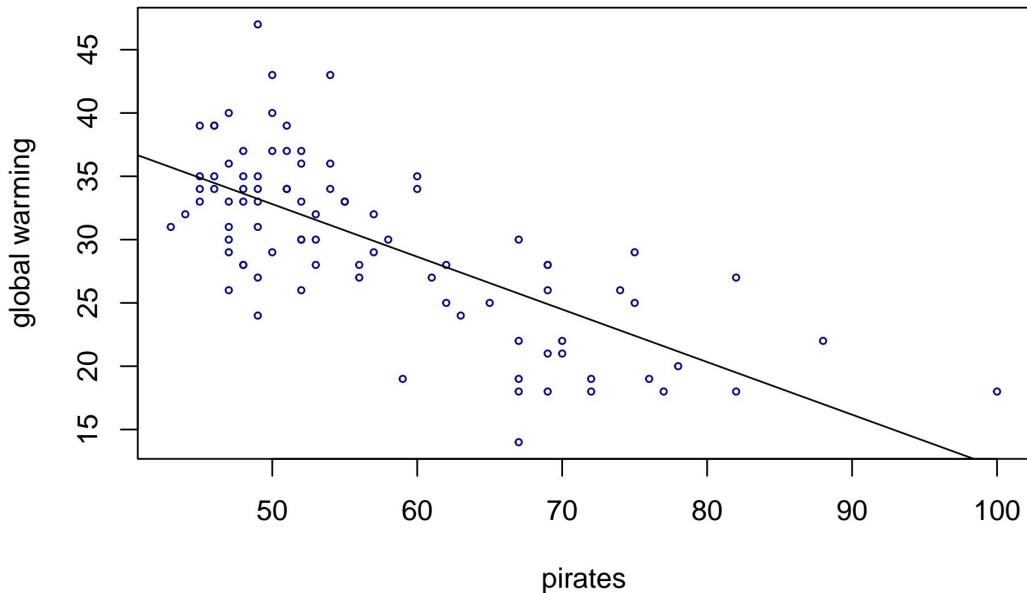
No questions assigned to the following page.

```

plot(pirate$pirates, pirate$global.warming, cex = 0.5, col = "blue4",
      xlab = "pirates", ylab = "global warming",
      main = "Google trend of pirate and gobal warming")
abline(pirate.lm)

```

**Google trend of pirate and gobal warming**



```

cor(pirate$pirates, pirate$global.warming)

## [1] -0.7097726

summary(pirate.lm)

##
## Call:
## lm(formula = global.warming ~ pirates, data = pirate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -11.7361  -3.3770  -0.0522   3.1548  13.7794 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 53.59517   2.57293  20.830 < 2e-16 ***
## pirates     -0.41581    0.04399  -9.452 4.81e-15 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.824 on 88 degrees of freedom

```

No questions assigned to the following page.

```

## Multiple R-squared:  0.5038, Adjusted R-squared:  0.4981
## F-statistic: 89.34 on 1 and 88 DF,  p-value: 4.814e-15

```

### The Failure of Google Flu Trend

Google Flu Trend was first launched in 2008 (The Google research paper: Detecting influenza epidemics using search engine query data). It was widely recognized an exciting event in the big data application In the 2009 flu pandemic, Google Flu Trends tracked information about flu in the United States. In February 2010, the CDC (the U.S. Centers for Disease Control and Prevention) identified influenza cases spiking in the mid-Atlantic region of the United States. However, Google's data of search queries about flu symptoms was able to show that same spike two weeks prior to the CDC report being released.

The model was initially based on the flu data from 2003-2008. Google Flu Trend prediction performs well at the beginning. However, it's been wrong since August 2011. The subsequent report continuously overestimates the flu prevalence. And now Google Flu Trends is no longer publishing current estimates of Flu based on search patterns.

In a Science article, a team of researchers described Google Flu Trend as "Big Data Hubris":

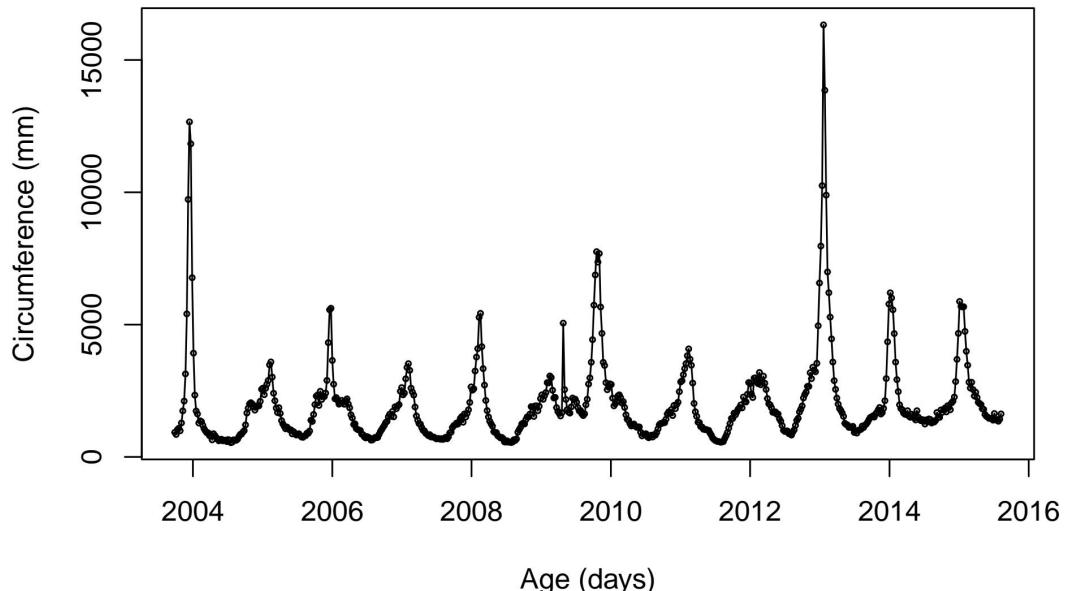
"The core challenge is that most big data that have received popular attention are not the output of instruments designed to produce valid and reliable data amenable for scientific analysis."

```

flu <- read.csv("flu.csv", stringsAsFactors = FALSE)
flu$date <- as.Date(flu$date)

plot(flu$date, flu$California,
      type = "o", cex = 0.4,
      xlab="Age (days)",
      ylab="Circumference (mm)" )

```



No questions assigned to the following page.

Reference and further reading:

- The Parable of Google Flu: Traps in Big Data Analysis
- Google Flu Trends' Failure Shows Good Data > Big Data
- Google Flu Trend

No questions assigned to the following page.

## LOESS: Local Polynomial Regression Fitting

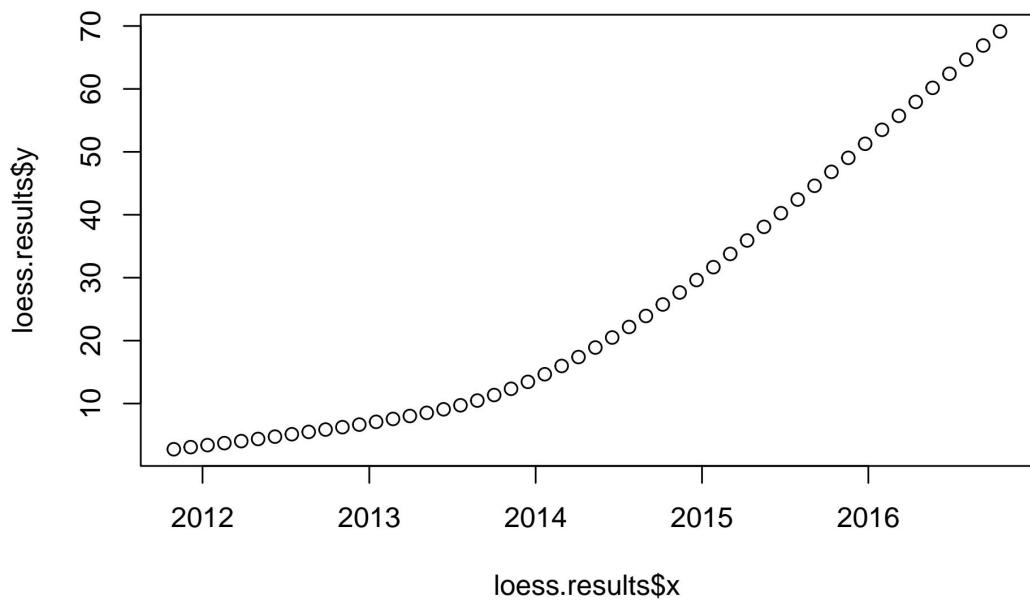
There are several options in R for fitting a loess.

The function `loess.smooth()` returns a list with the smoothed data coordinates:

```
loess.results <- loess.smooth(x = data_science$week, y = data_science$r)
loess.results

## $x
## [1] "2011-10-30" "2011-12-06" "2012-01-12" "2012-02-18" "2012-03-26"
## [6] "2012-05-02" "2012-06-08" "2012-07-15" "2012-08-21" "2012-09-27"
## [11] "2012-11-03" "2012-12-10" "2013-01-16" "2013-02-22" "2013-03-31"
## [16] "2013-05-07" "2013-06-13" "2013-07-20" "2013-08-26" "2013-10-02"
## [21] "2013-11-08" "2013-12-15" "2014-01-21" "2014-02-27" "2014-04-05"
## [26] "2014-05-12" "2014-06-18" "2014-07-25" "2014-08-31" "2014-10-07"
## [31] "2014-11-13" "2014-12-20" "2015-01-26" "2015-03-04" "2015-04-10"
## [36] "2015-05-17" "2015-06-23" "2015-07-30" "2015-09-05" "2015-10-12"
## [41] "2015-11-18" "2015-12-25" "2016-01-31" "2016-03-08" "2016-04-14"
## [46] "2016-05-21" "2016-06-27" "2016-08-03" "2016-09-09" "2016-10-16"
##
## $y
## [1] 2.736377 3.072615 3.392789 3.708313 4.030601 4.371064 4.741115
## [8] 5.123584 5.494673 5.865706 6.248098 6.653264 7.092621 7.554054
## [15] 8.022257 8.522621 9.080914 9.722903 10.474354 11.353310 12.346029
## [22] 13.448366 14.657861 15.972053 17.388485 18.901316 20.494740 22.166603
## [29] 23.916373 25.743519 27.647511 29.627548 31.675587 33.776878 35.916586
## [36] 38.079874 40.251906 42.420154 44.608423 46.820078 49.046064 51.277326
## [43] 53.504812 55.719529 57.936788 60.169443 62.411485 64.656900 66.899678
## [50] 69.133807
plot(loess.results$x, loess.results$y)
```

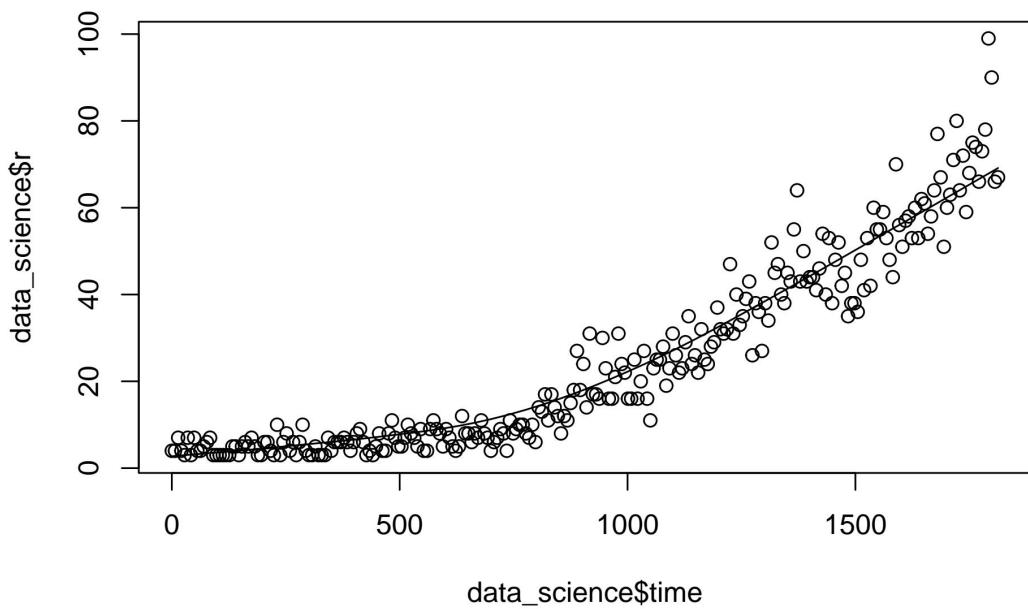
No questions assigned to the following page.



Though `loess.smooth()` is convenient for plotting the smoothed fit, there is also `scatter.smooth()`, which prints both the scatter plot and the smoothed fit with just one line of code.

```
scatter.smooth(x=data_science$time, y=data_science$r)
```

No questions assigned to the following page.

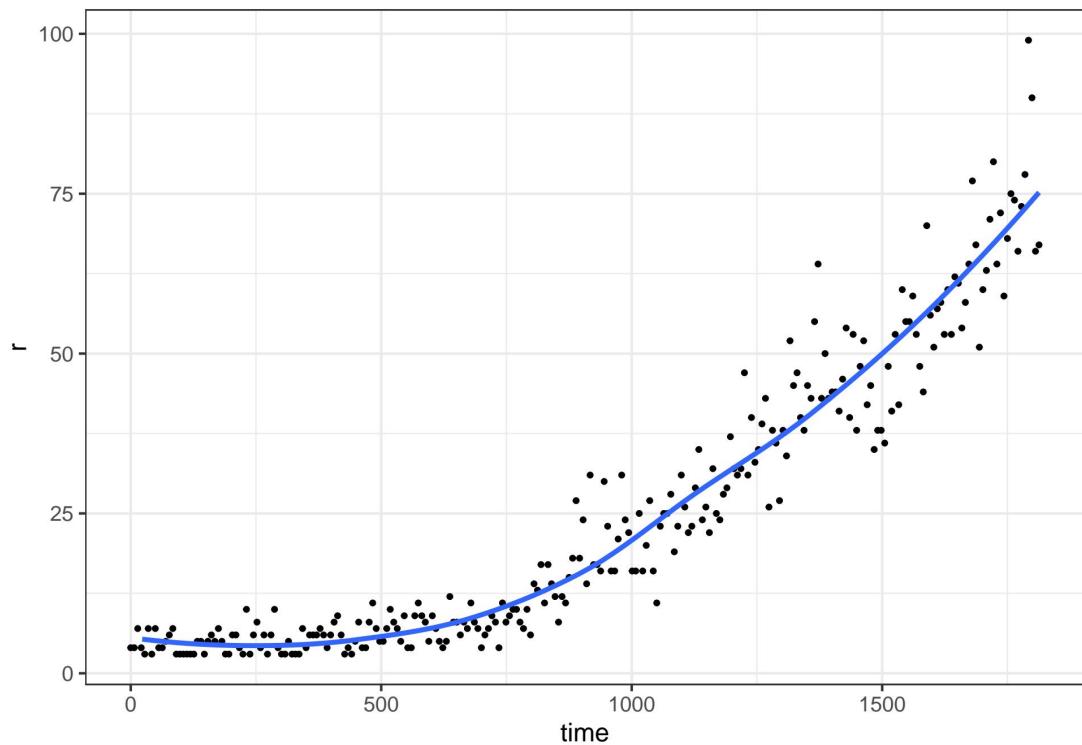


Using ggplot2, we have

```
data_science %>% ggplot(aes(
  x = time, y = r
)) + geom_point(size = 0.7) +
  geom_smooth(
    method = 'loess', span = 2/3,
    formula = y ~ poly(x, degree = 1),
    se = F) +
  theme_bw()

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

No questions assigned to the following page.



To do prediction using a LOESS model, there is function `loess()`, which has similar usage as the `lm()` function. Notice the default smoothing parameter for `loess.smooth(span = 2/3, degree = 1)` and `loess(span = 0.75, degree = 2)` are different.

```

loess.fitted <- loess(r~time, data = data_science)
summary(loess.fitted)

## Call:
## loess(formula = r ~ time, data = data_science)
##
## Number of Observations: 260
## Equivalent Number of Parameters: 4.35
## Residual Standard Error: 5.464
## Trace of smoother matrix: 4.73 (exact)
##
## Control settings:
##   span     : 0.75
##   degree   : 2
##   family   : gaussian
##   surface  : interpolate      cell = 0.2
##   normalize: TRUE
##   parametric: FALSE
##   drop.square: FALSE
predict(loess.fitted, data.frame(time = c(1000, 1500)))

## [1] 20.79013 50.11676

```

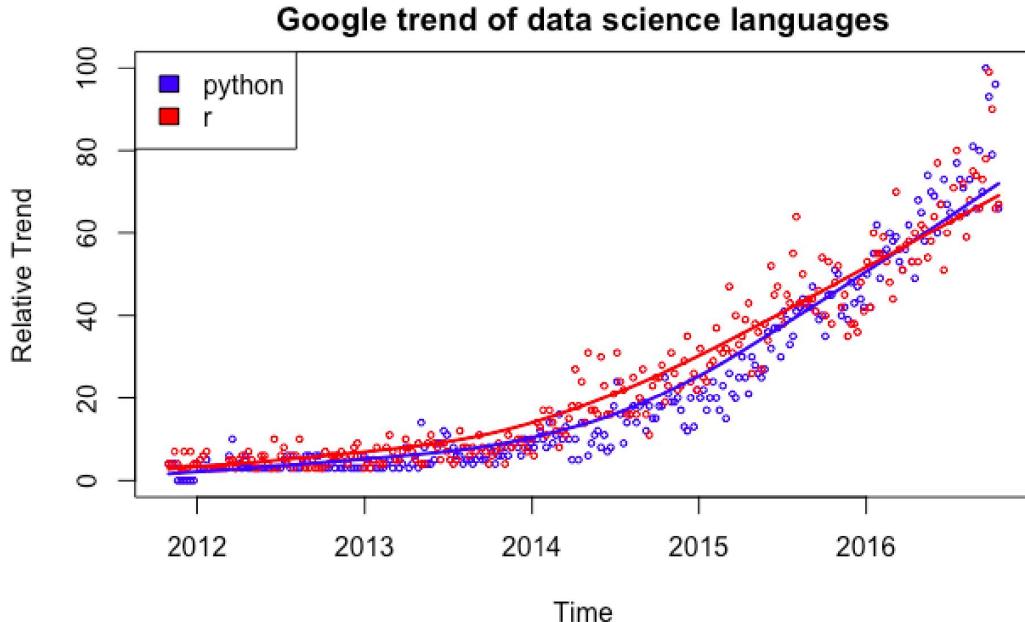
Question assigned to the following page: [4](#)

## Exercise 2

Follow the steps below to create a plot:

- (1) Plot a scatter plot of `week` versus `r` and `week` versus `python` in the same pane; distinguish `r` from `python` by color.
- (2) Overlay the scatter plots with a LOESS smoothing line for both `r` and `python` appropriately matched colors.
- (3) Make sure that a legend is included in the plot.

Your plot should look something like the following:



```
# Insert your code here
data_science %>%
  pivot_longer (c("python", "r")) %>%
  ggplot(aes(x= week, y =value, color = name)) +
  geom_point() + geom_smooth(method = "loess", formula = y~x) +
  labs(x = "Date",y = "Search") +
  theme(legend.position = c(0.1, 0.8))
```

Question assigned to the following page: [4](#)

