Lab 07

Stat 131A

Welcome to lab 7! In this lab, you will:

- Learn how to do LOESS regression.
- Explore some visualization methods for data sets with categorical variables.

LOESS: Local Polynomial Regression Fitting

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
```

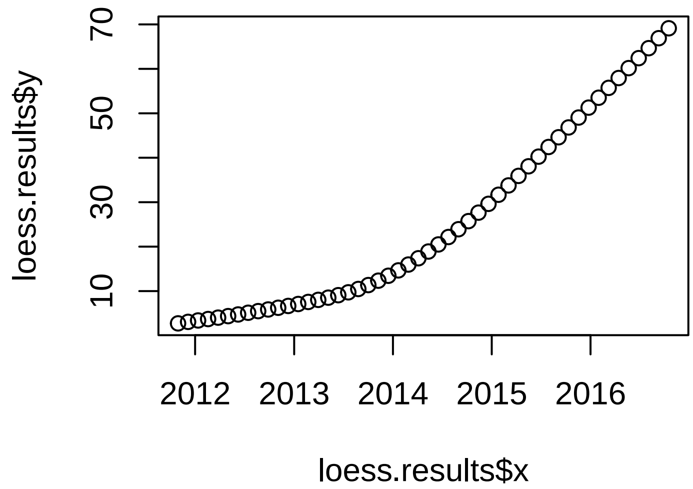
There are several options in `R` for fitting a loess.

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
```

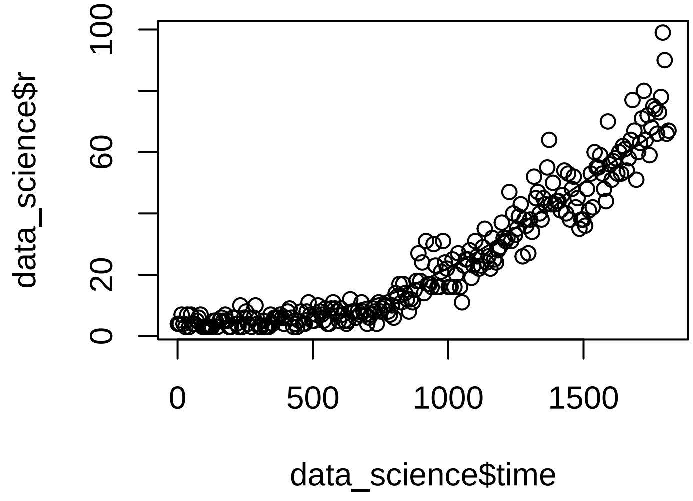
```
## $x
##  [1] "2011-10-30" "2011-12-06" "2012-01-12" "2012-02-18" "2012-03-26"
##  [6] "2012-05-02" "2012-06-08" "2012-07-15" "2012-08-21" "2012-09-27"
## [11] "2012-11-03" "2012-12-10" "2013-01-16" "2013-02-22" "2013-03-31"
## [16] "2013-05-07" "2013-06-13" "2013-07-20" "2013-08-26" "2013-10-02"
## [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)
```



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.

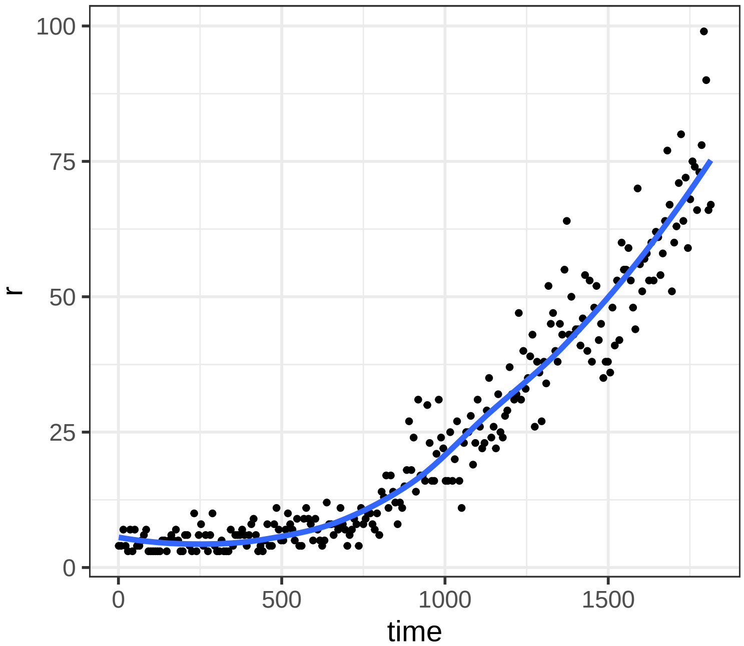
```
scatter.smooth(x=data_science$time,y=data_science$r)
```



Using `ggplot2`, we have

```
library(tidyverse)
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()
```



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] 20.73465 50.04658
```

Exercise 1

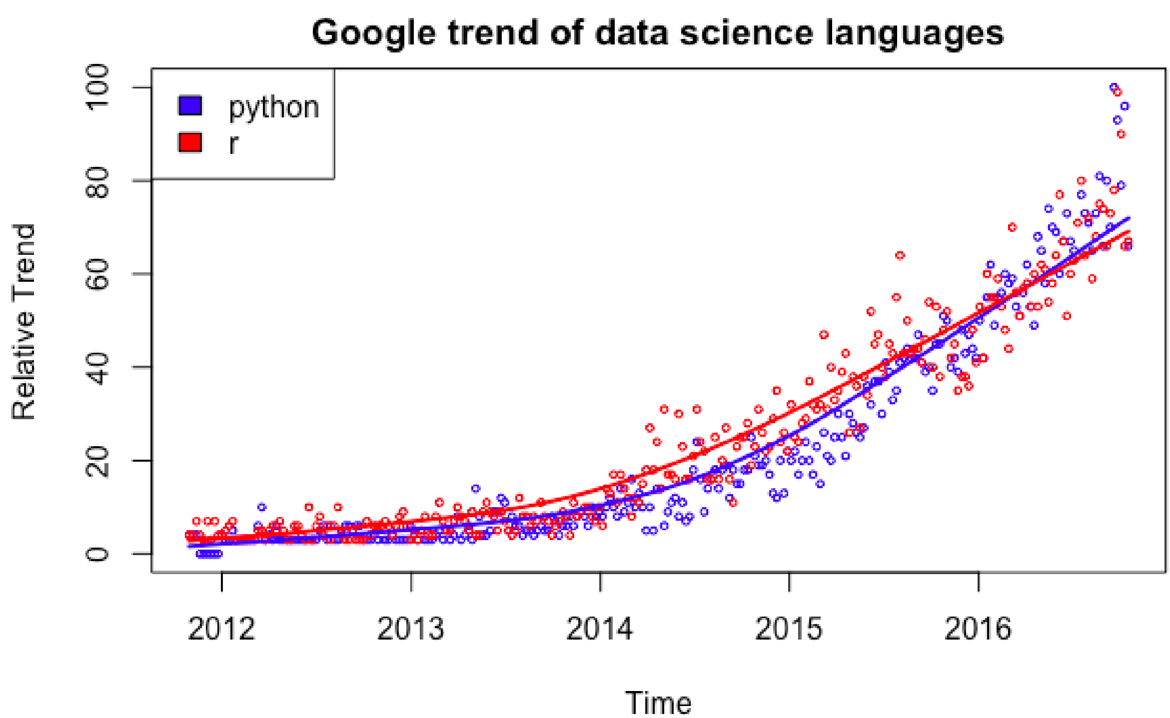
Follow the steps below to create a plot:

(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.

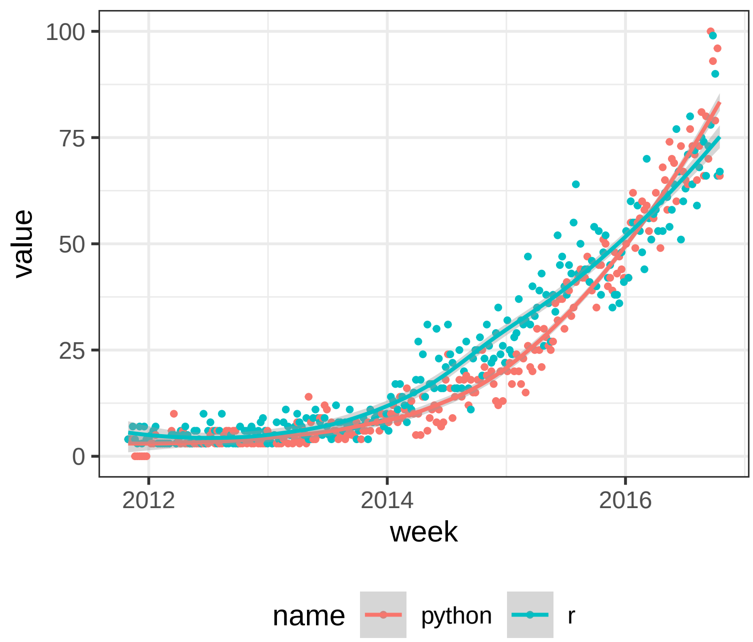
Your plot should look something like the following:




```
library(tidyverse)
library(ggplot2)
# Insert your code for plotting here
data_science %>% pivot_longer(
  c('python','r')
  )%>% ggplot(aes(x=week, y=value, color=name))+
  geom_point(size=0.7)+
  geom_smooth(method="loess",span=2/3,
              fomula=y~x,size =0.7)+theme_bw()+
  theme(legend.position="bottom")
```

```
## Warning: Ignoring unknown parameters: fomula
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Multivariate Data visualization - Online news popularity

The data stored in `OnlineNewsPopularity.csv` includes several variables describing articles published by Mashable over a period of two years. This dataset contains 49 variables for each news post, including

- `weekday`: Days of week. Mon, Tue, Wed, etc.
- `channel`: Channel. Tech, Entertainment, Business, etc.
- `shares`: Number of shares.
- `num_imgs`: Number of images.
- `num_videos`: Number of videos.
- `n_tokens_title`: : Number of words in the title.
- `num_hrefs`: Number of links

Read in data.

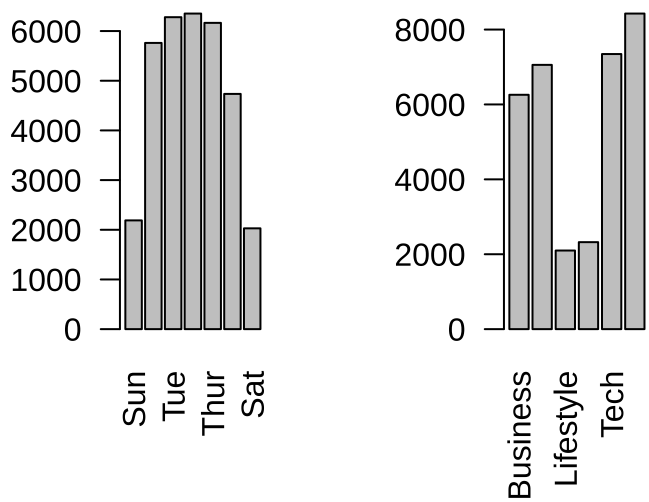
```
popul <- read.csv("OnlineNewsPopularity.csv")
popul$weekday <- factor(popul$weekday, c("Sun", "Mon", "Tue", "Wed", "Thur", "Fri", "Sat"))
head(popul[,c("weekday", "channel", "shares", "num_imgs")])
```

```
##   weekday       channel shares num_imgs
## 1     Mon Entertainment    593        1
## 2     Mon      Business    711        1
## 3     Mon      Business   1500        1
## 4     Mon Entertainment   1200        1
## 5     Mon          Tech    505       20
## 6     Mon          Tech    855        0
```

Exercise 2

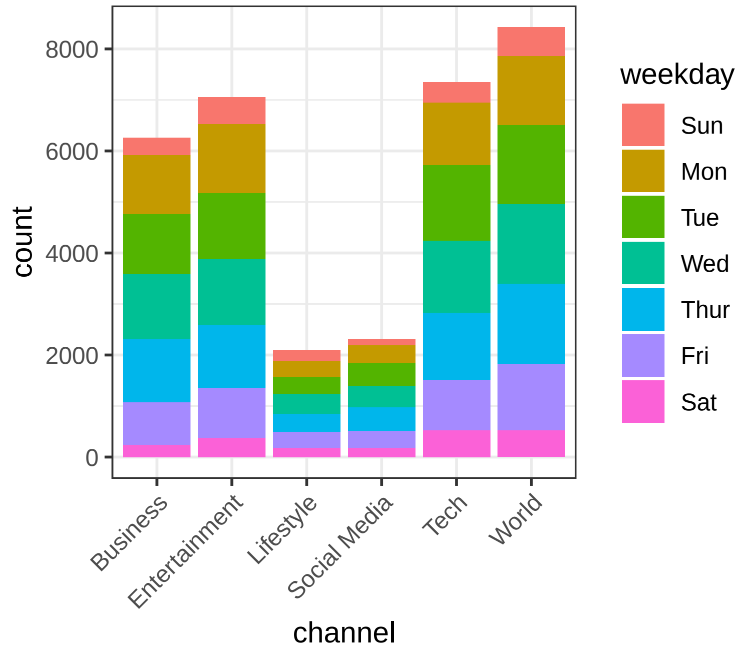
(a) Construct side-by-side bar plots for `weekdays` and `channels` the using `barplot()` and `table()` functions. Rotate the horizontal axis labels using argument `las = 2` in `barplot()`. *Hint*: use `par(frow...)`.

```r
# Insert your code for plotting here
par(mfrow=c(1,2))
barplot(table(popul$weekday),las=2)
barplot(table(popul$channel),las=2)
```



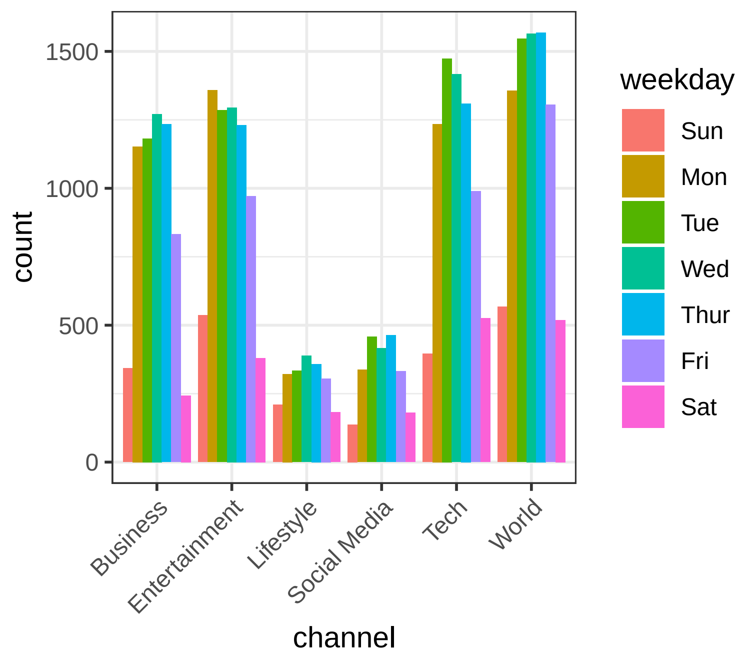
(b) Create a contingency table for `weekdays` and `channels` using `table()`. Use `barplot()` to visualize the relationship between two categorical variables.

```r
# Insert your code for plotting here
popul[,c('weekday','channel')] %>% ggplot(
  aes(x=channel,fill=weekday))+
  geom_bar()+
  theme_bw()+
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



(c) Use the contingency table you created in (b) to get separate bar plots for days of the week. *Hint*: use `beside = TRUE`.

```
# Insert your code for plotting here
popul[,c('weekday','channel')] %>% ggplot(
  aes(x=channel,fill=weekday))+
  geom_bar(position="dodge")+
  theme_bw()+
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



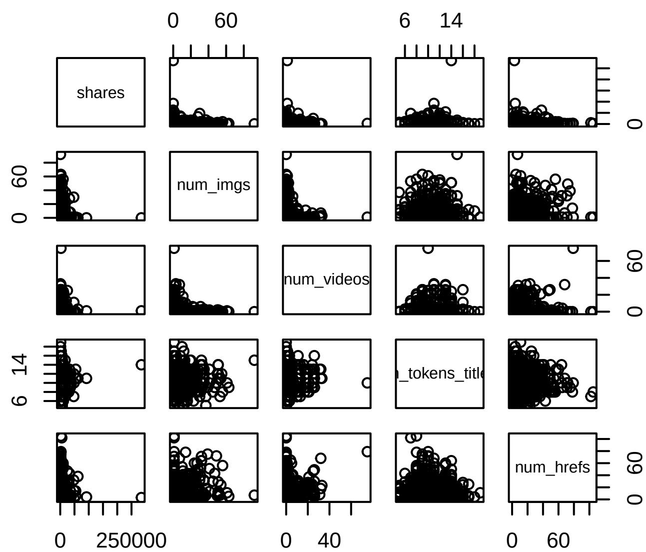
Exercise 3

The following script takes a subset of 2000 rows from the news popularity dataset.

```
vars <- c("weekday", "channel", "shares", "num_imgs", "num_videos", "n_tokens_title", "num_hrefs")
sample.idx <- sample(nrow(popul), 2000)
popul.subset <- popul[sample.idx, vars]
```

(a) Generate a matrix of scatter plots for all variable pairs in `popul.subset` that are of class `numeric` using the `pairs()` function.

```
# Insert your code for plotting here
pairs(popul.subset[,sapply(popul.subset,is.numeric)])
```

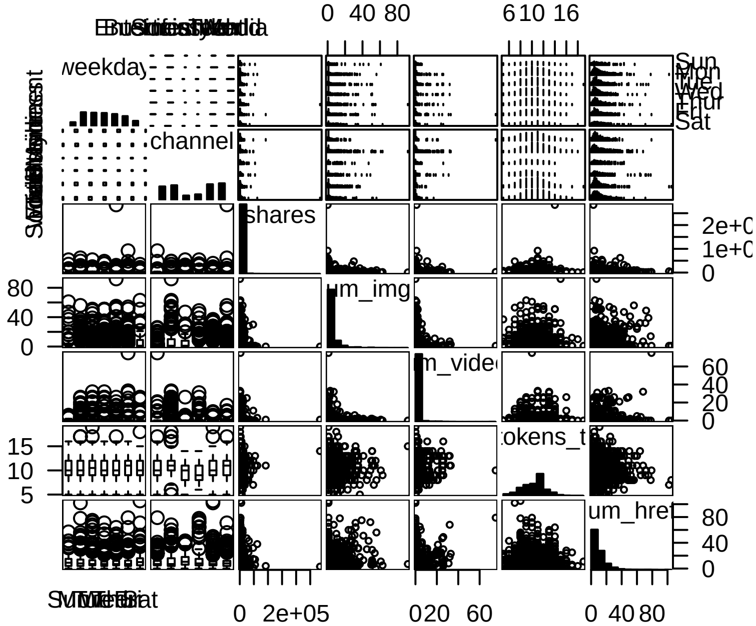


(b) Generate a matrix of scatter plots for all variable pairs in `popul.subset` using the `gpairs()` function in the `gpairs` package.

```
library(gpairs)
# Insert your code for plotting here
popul.subset$channel <- factor(popul.subset$channel)
gpairs(popul.subset)
```

```
## Loading required package: grid

## Loading required package: lattice
```



weekday
channel
shares
um_img
m_vide
okens_t
um_href
Sun
Mon
Tue
Wed
Thur
Fri
Sat
0 40 80
6 10 16
0 2e+05
0 20 60
0 40 80
15
10
5
60
40
20
0

For fun (not required)

(a) Plot the alluvial plot for `weekday` and `channel` using `alluvial()` in the `alluvial` package.

```
library(alluvial)
# Insert your code for plotting here
```

(b) Plot the mosaic plot plot for `weekday` and `channel` using `mosaicplot()`.

```
# Insert your code for plotting here
```