Student Led R Session 03

Felix Mueller

3/21/2017

Topics of today

- writing / reading files with specific format
- CF 3 recommendation engines
- ggplot2
- handling NAs
- Feature derivation
- kNN
- naive Bayes
- trees
- Random Forests
- Logistic regression

writing / reading files with specific format

```
print(help(write.csv))
```

write.table(x, file = “”, append = FALSE, quote = TRUE, sep = " “, eol =”n“, na =”NA“, dec =”.“, row.names = TRUE, col.names = TRUE, qmethod = c(”escape“,”double“), fileEncoding =”“)

CF 3 recommendation engines I

Formulas from class (Introduction to Recommendation Engines, slide 59):

user similarity (Pearson correlation)

$$\mathrm{simil}(x,y) = \frac{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum\limits_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

rating prediction (weighted sum)

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \mathrm{simil}(u,u')(r_{u',i} - \bar{r_{u'}})$$

$$k = 1 / \sum_{u' \in U} |\mathrm{simil}(u,u')|$$

CF 3 recommendation engines II

```r
victoria <- critics %>% filter(User=="Victoria") %>%
  select(-User) %>% as.integer() #from 3
similarity <- critics %>% filter(User != "Victoria"
  ) %>% select(-User) %>% t() %>% cor(.,victoria,use=
  "pairwise.complete.obs",method="pearson") #from 4
avg_rating = critics %>% filter(User!="Victoria"
  ) %>% select(-User)
avg_rating=apply(avg_rating,1,function(x)mean(x,na.rm=T))
avg_victoria = critics %>% filter(User=="Victoria") %>%
  select(-User) %>% t() %>% mean(.,na.rm = T)
```

CF 3 recommendation engines III

```
critics_no_victoria = critics %>% filter(User !=
  "Victoria") %>% select(-User)
pred_by_movie = vector()
for (i in seq(1:ncol(critics_no_victoria))){
  pred_by_movie[i]=sum(similarity * (
    critics_no_victoria[i]-avg_rating),na.rm=T)
}
#let's normalise the values
rating_calculated <- avg_victoria + (
  (1/sum(abs(similarity))) * pred_by_movie)
movie_names <- names(critics[,-1])
rating_calculated = as.data.frame(cbind(
  rating_calculated,movie_names))
```

CF 3 recommendation engines IV

```
#exclude movies she has already watched
critics_victoria = critics %>% filter(User == "Victoria"
  ) %>% select(-User) %>% t() %>% is.na()
rating_calculated %>% filter(critics_victoria) %>%
  arrange(.,desc(rating_calculated)) %>% top_n(5,
  rating_calculated)
```

```
##   rating_calculated         movie_names
## 1   3.7917013044215          The Matrix
## 2  3.50776533175371        Forrest Gump
## 3  3.33118834864677     The Sixth Sense
## 4  3.11491825315719 Shakespeare in Love
## 5   2.912451322866 5        Blade Runner
```

Possible kinds of feature engineering

- Adding values
- Multiplying values
- Decompose Categorical Attributes
- Decompose a Date-Time
- Categories to numeric and the other way around
- PCA
- ML models predicting as a variable (i.e. Bagged trees' predictions)

More Information: `http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/`

Decompose a Date-Time

```
median(df$date_recorded)
```

```
## [1] "2012-10-10"
```

```
max(df$date_recorded)
```

```
## [1] "2013-12-03"
```

```
min(df$date_recorded)
```

```
## [1] "2002-10-14"
```

```
df$new_date = if_else(df$date_recorded>median
   (df$date_recorded),"recent", "not recent")
head(df$new_date)
```

```
## [1] "not recent" "not recent" "not recent" "recent"
## [6] "not recent"
```

Handling NAs

- most similar value
- not use the feature
- create a new category for it
- not recommended: exclude instances from sample
- not recommended: do nothing

Handling NAs: applied I

```
#check for NAs
names = df[,apply(df, 2, function(x)
  any(is.na(x)))] %>% names()
apply(df[,names], 2, function(x) sum(is.na(x)))
```

```
##              funder         installer        subvillage
##                3269              3285               333
## scheme_management       scheme_name            permit
##                3476             25424              2768
```

```
#we handle scheme_name NAs as a new category
#because NAs reflect about 50% of the data
index = sapply(df$scheme_name,is.na)
levels(df$scheme_name) = c(levels(df$scheme_name),
                           "unknown")
df$scheme_name[index] = "unknown"
```

Handling NAs: applied II

```
get_mode <- function(x) {
  unique_x <- unique(x)
  unique_x[which.max(tabulate(match(x, unique_x)))]
}
#update names
names = df[,apply(df, 2, function(x)
  any(is.na(x)))] %>% names()
names
```

```
## [1] "funder"            "installer"         "subvillage'
## [4] "public_meeting"    "scheme_management" "permit"
```

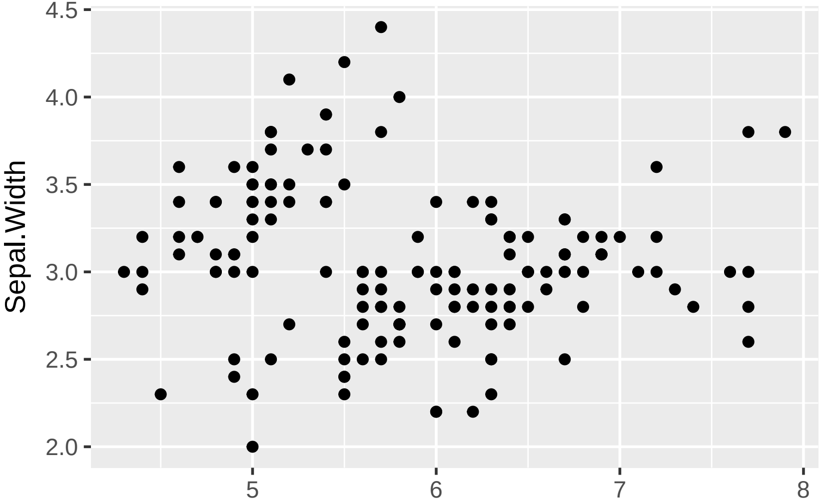
Handling NAs: applied III

```
df = data.frame(df)
for (name in names){
  mode = get_mode(df[,name])
  index = sapply(df[,name],is.na)
  levels(df[,name]) = c(levels(df[,name]),mode)
  df[index,name] = mode
}
#finished NA handling
print(df[,apply(df, 2, function(x) any(is.na(x)))]
      %>% names())
```

```
## character(0)
```

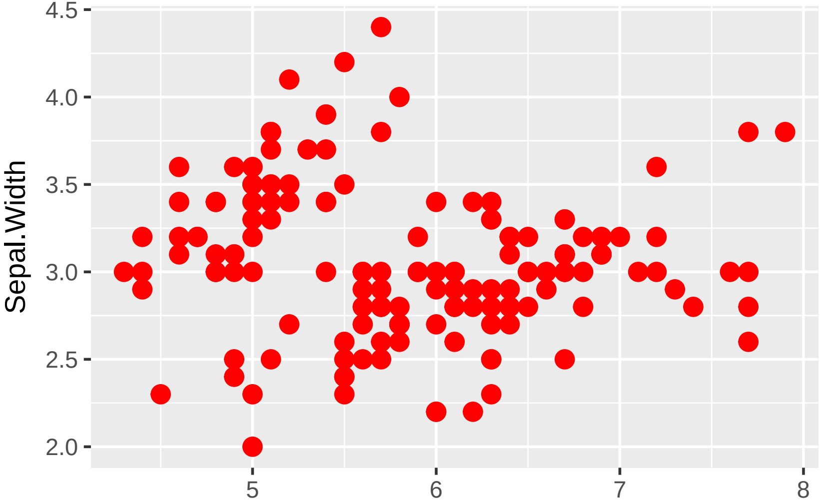
ggplot2 I

```
ggplot(data = iris, mapping =
  aes(x=Sepal.Length,y=Sepal.Width))+
  geom_point()
```



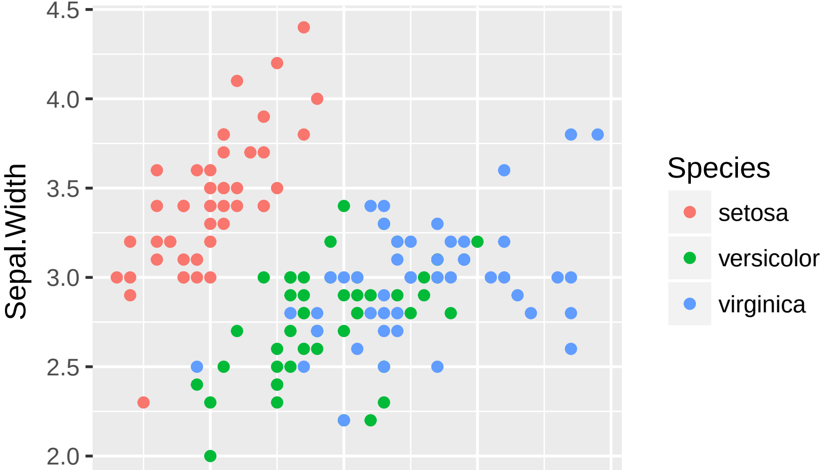
ggplot2 II

```
ggplot(data = iris, mapping =
  aes(x=Sepal.Length,y=Sepal.Width))+
  geom_point(colour = I('red'), size = 3)
```



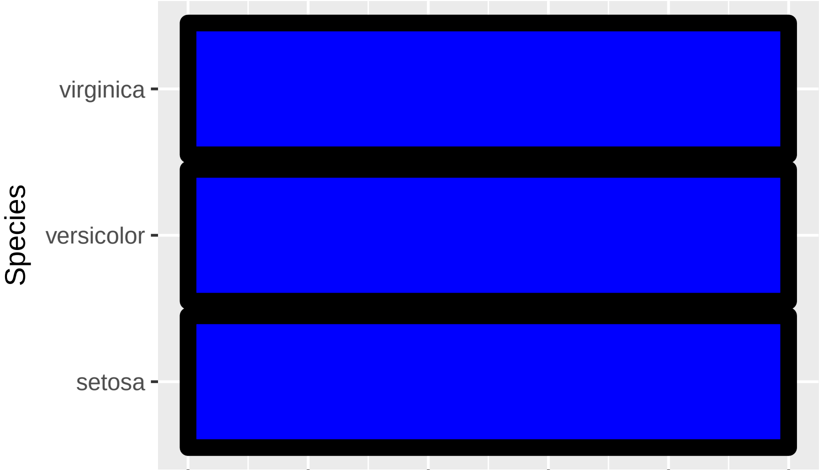
ggplot2 III

```
ggplot(data = iris, mapping =
  aes(x=Sepal.Length,y=Sepal.Width,
  color=Species))+
  geom_point()
```



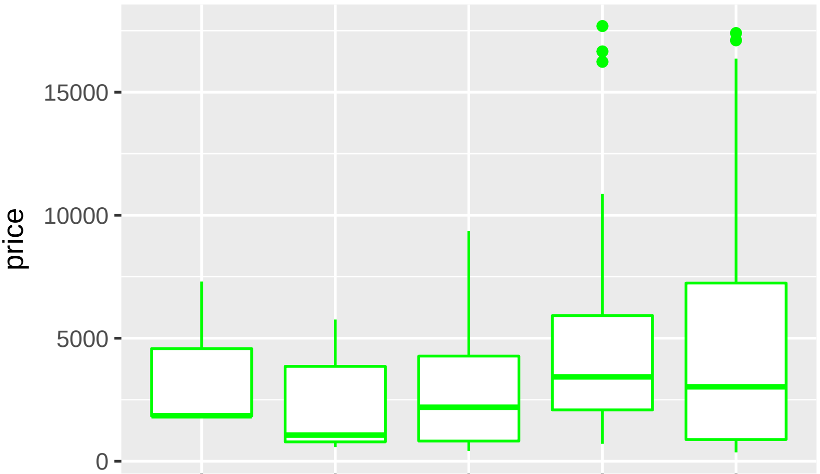
ggplot2 IV

```r
ggplot(data = iris, mapping =
  aes(x=Species))+
  geom_bar(colour=I("black"),fill=I("blue"),size=3)+
  coord_flip()
```



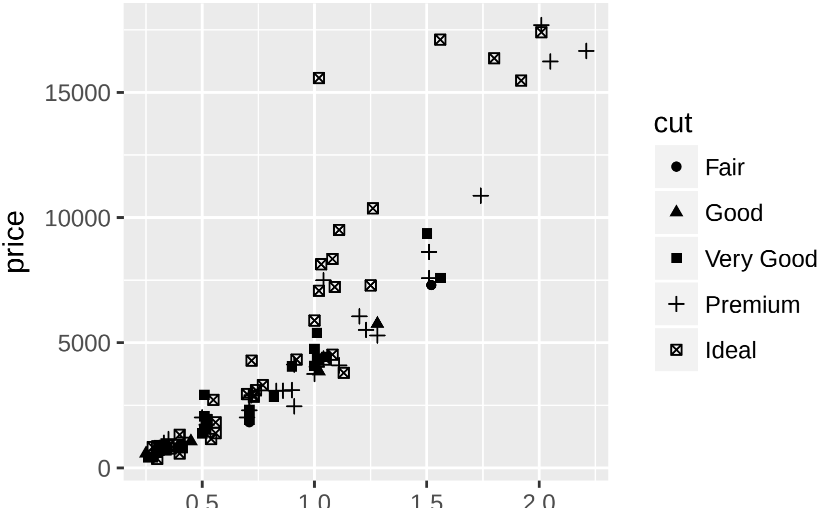
ggplot2 V

```
d = diamonds %>% sample_n(100)
ggplot(data = d, mapping =
  aes(x=cut,y=price))+
  geom_boxplot(colour=I("green"))
```



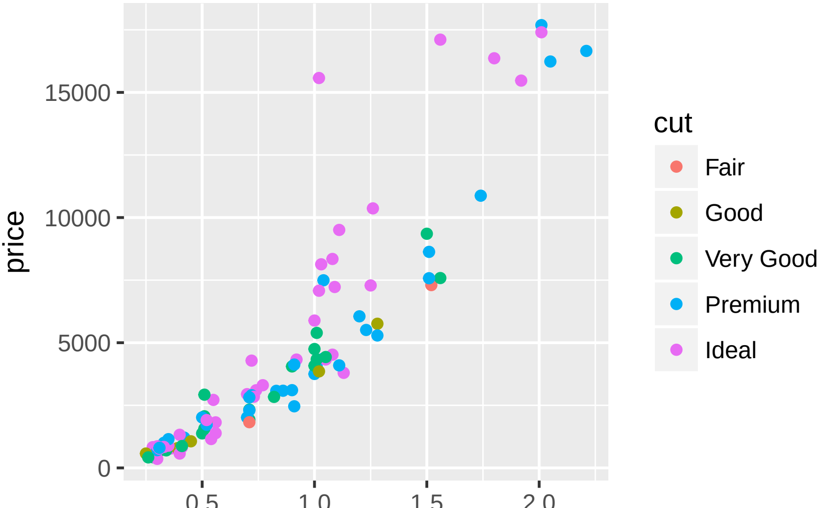
ggplot2 VI

```
ggplot(data = d, mapping =
  aes(carat,price,shape=cut))+
  geom_point()
```



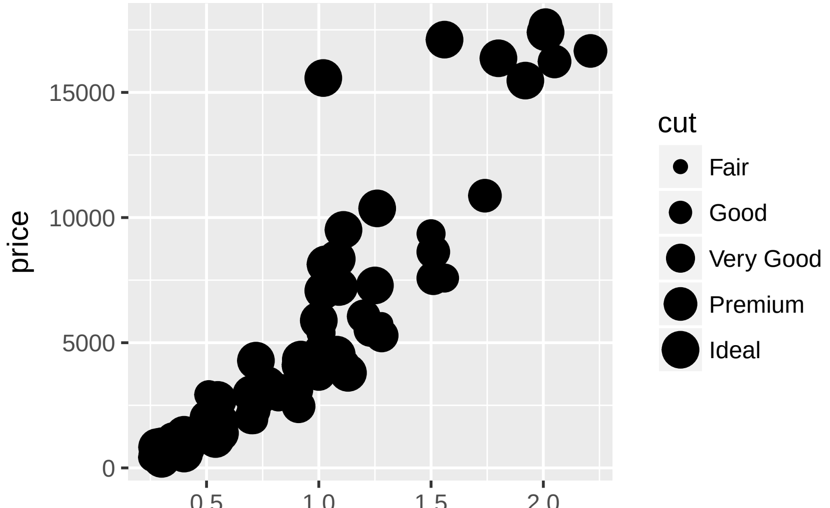
ggplot2 VII

```
ggplot(data = d, mapping =
  aes(carat,price,colour=cut))+
  geom_point()
```



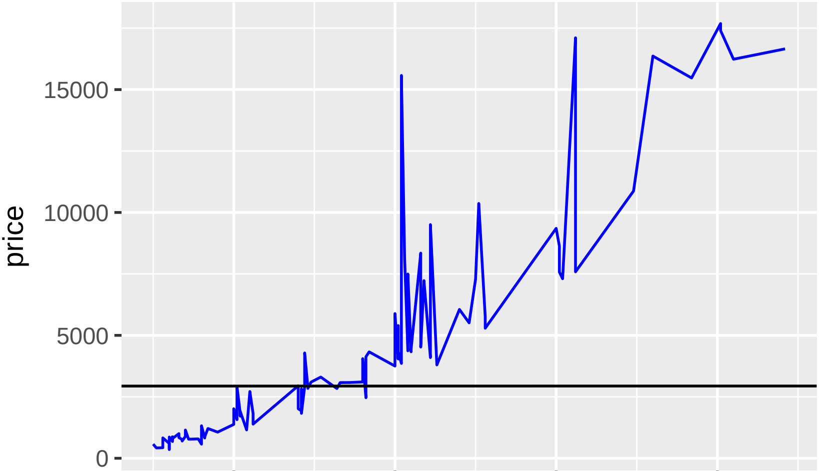
ggplot2 VIII

```
ggplot(data = d, mapping =
  aes(carat,price,size=cut))+
  geom_point()
```



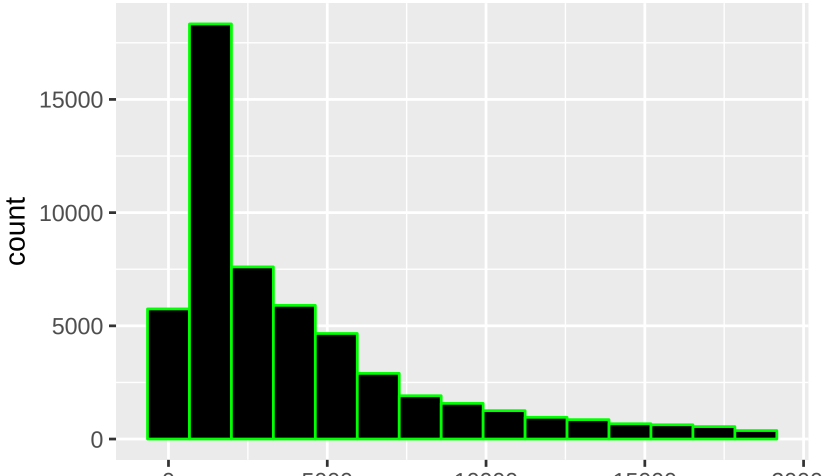
ggplot2 IX

```
ggplot(data = d, mapping =
  aes(carat,price))+
  geom_line(colour=I("blue"))+
  geom_abline(intercept = median(d$price))
```



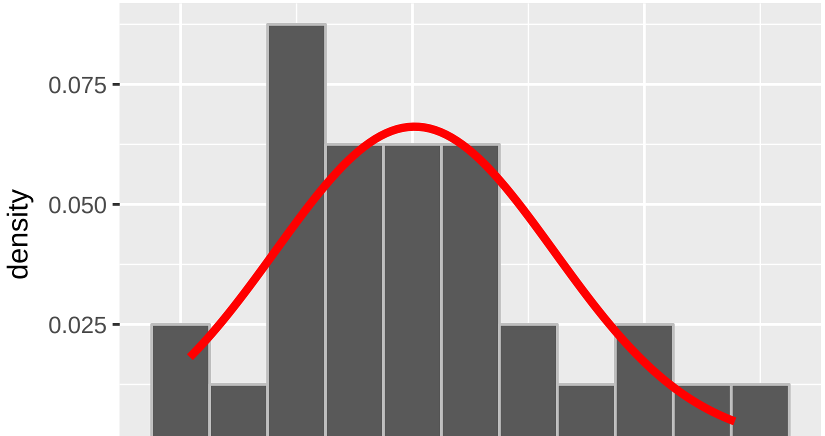
ggplot2 X

```
ggplot(data = diamonds, mapping =
  aes(price))+
  geom_histogram(colour=I("green"),fill=I("black"),
  bins = 15)
```



ggplot2 XI

```
ggplot(data=mtcars,aes(x=mpg))+
  geom_histogram(aes(y=..density..),binwidth=2.5,
  colour=I("gray")) +
  stat_function(fun=dnorm,args=list(mean=mean(mtcars$mpg),
  sd=sd(mtcars$mpg)),colour=I("red"),size=1.5)
```



naive Bayes

Computes the conditional probabilities of an event

Good if:

- missing data
- tons of features

Watch out if:

- highly correlated predictors
- use log probabilites for many features
- zero observations problem

trees

creates set of rules based on posterior & prior probabilities

Good if:

- categorical data
- interpretation necessary

Watch out if:

- many numeric attributes
- trees overfit your data –> pruning!

Random Forest

creates several decorrelated decision trees (usually > 100)

Good if:

- categorical data
- correlated predictors
- big dataset

Watch out if:

- many numeric attributes

Logistic regression

estimates regression coefficients with Maximum Likelihood function

Good if:

- instances k$>>$predictors p
- used for feature selection with $\sim$20-50 predictors

Watch out if:

- too many predictive variables (10k $<$ p)