Machine Learning: Regression in R

Jan-Philipp Kolb

01 Juni, 2019

Why a part on simple regression

- OLS can be seen as a simple machine learning technique
- Some other machine learning concepts are based on regression (e.g. regularization).
- We would like to remind you how simple regression works in R.
- We also want to show the constraints
- In a next step we will learn, how to coop with these constraints

VARIABLES OF THE MTCARS DATASET

Help for the mtcars dataset:

?mtcars

- ▶ mpg - Miles/(US) gallon
- ▶ cyl - Number of cylinders
- ▶ disp - Displacement (cu.in.)
- ▶ hp - Gross horsepower
- ▶ drat - Rear axle ratio
- ▶ wt - Weight (1000 lbs)
- ▶ qsec - 1/4 mile time
- ▶ vs - Engine (0 = V-shaped, 1 = straight)
- ▶ am - Transmission (0 = automatic, 1 = manual)
- ▶ gear - Number of forward gears
- ▶ carb - Number of carburetors

Dataset mtcars

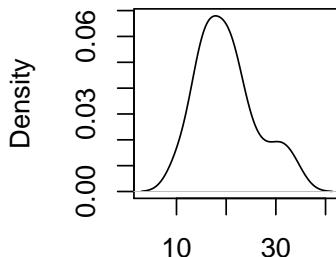|	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0			
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0			
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1			
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1			
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0			
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1			
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0			
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1			
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1			
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1			
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1			
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0			
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0			
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0			
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0			
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0			

Distributions of two variables of MTCARS

```r
par(mfrow=c(1,2))
plot(density(mtcars$wt)); plot(density(mtcars$mpg))
```

density.default(x = mtcars$density.default(x = mtcars$
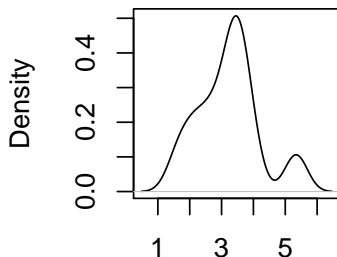
N = 32   Bandwidth = 0.3455   N = 32   Bandwidth = 2.477

A simple regression model

Dependent variable - miles per gallon (mpg)

Independent variable - weight (wt)

```
m1 <- lm(mpg ~ wt,data=mtcars)
m1

##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)            wt
##      37.285        -5.344
```

```
summary(m1)

##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
##
## Residual standard error: 3.046 on 30 degrees of freedom
```

The model formula

Model without intercept

```r
m2 <- lm(mpg ~ - 1 + wt,data=mtcars)
summary(m2)$coefficients
```

```
##    Estimate Std. Error  t value    Pr(>|t|)
## wt 5.291624  0.5931801 8.920771 4.55314e-10
```

Adding further variables

```r
m3 <- lm(mpg ~ wt + cyl,data=mtcars)
summary(m3)$coefficients
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 39.686261   1.7149840 23.140893 3.043182e-20
## wt          -3.190972   0.7569065 -4.215808 2.220200e-04
## cyl         -1.507795   0.4146883 -3.635972 1.064282e-03
```

THE COMMAND `AS.FORMULA`

```
?as.formula

class(fo <- mpg ~ wt + cyl)

## [1] "formula"

# The formula object can be used in the regression:
m3 <- lm(fo,data=mtcars)
```

Further possibilities to specify the formula

Take all available predictors

```
m3_a<-lm(mpg~.,data=mtcars)
```

Interaction effect

```
# effect of cyl and interaction effect:
m3a<-lm(mpg~wt*cyl,data=mtcars)

# only interaction effect:
m3b<-lm(mpg~wt:cyl,data=mtcars)
```

Take the logarithm

```
m3d<-lm(mpg~log(wt),data=mtcars)
```

THE COMMAND SETDIFF

- ▶ We can use the command to create a dataset with only the features, without the dependent variable

```
names(mtcars)
```

```
## [1]  "mpg"  "cyl"  "disp" "hp"   "drat" "wt"  "qsec" "vs"
## [11] "carb"
```

```
features <- setdiff(names(mtcars), "mpg")
features
```

```
## [1]  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
```

```
featdat <- mtcars[,features]
```

THE COMMAND `MODEL.MATRIX`

▶ With model.matrix the qualitative variables are automatically dummy encoded

```
?model.matrix
```

```
model.matrix(m3d)
```

```
##                     (Intercept)  log(wt)
## Mazda RX4                     1 0.9631743
## Mazda RX4 Wag                 1 1.0560527
## Datsun 710                    1 0.8415672
## Hornet 4 Drive                1 1.1678274
## Hornet Sportabout             1 1.2354715
## Valiant                       1 1.2412686
## Duster 360                    1 1.2725656
## Merc 240D                     1 1.1600209
## Merc 230                      1 1.1474025
## Merc 280                      1 1.2354715
## Merc 280C                     1 1.2354715
```

Model matrix (II)

▶ We can also create a model matrix directly from the formula and data arguments

▶ See Matrix::sparse.model.matrix for increased efficiency on large dimension data.

```r
ff <- mpg ~ log(wt):cyl
m <- model.frame(ff, mtcars)

(mat <- model.matrix(ff, m))
```

```
##                     (Intercept) log(wt):cyl
## Mazda RX4                     1    5.779046
## Mazda RX4 Wag                 1    6.336316
## Datsun 710                    1    3.366269
## Hornet 4 Drive                1    7.006964
## Hornet Sportabout             1    9.883772
## Valiant                       1    7.447612
## Duster 360                    1   10.180525
## Merc 240D                     1    4.640084
```

A MODEL WITH INTERACTION EFFECT

```r
# disp  - Displacement (cu.in.)
m3d<-lm(mpg~wt*disp,data=mtcars)
m3dsum <- summary(m3d)
m3dsum$coefficients

##                  Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept) 44.08199770 3.123062627 14.114990 2.955567e-14
## wt          -6.49567966 1.313382622 -4.945763 3.216705e-05
## disp        -0.05635816 0.013238696 -4.257078 2.101721e-04
## wt:disp      0.01170542 0.003255102  3.596022 1.226988e-03
```

Residual plot - model assumptions violated?

▶ We have model assumptions violated if points deviate with a pattern from the line

```
plot(m3,1)
```


Residuals vs Fitted
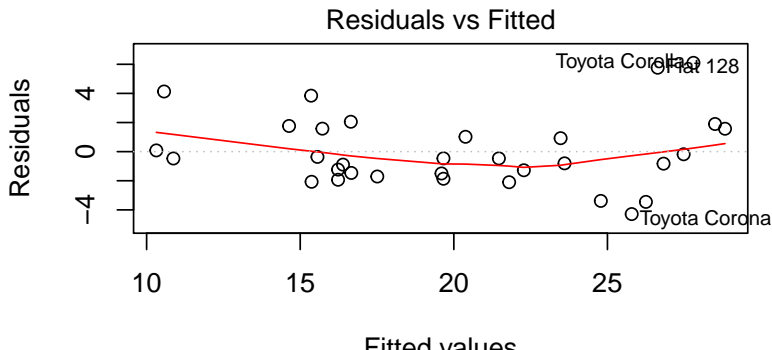
Residual plot

`plot(m3,2)`
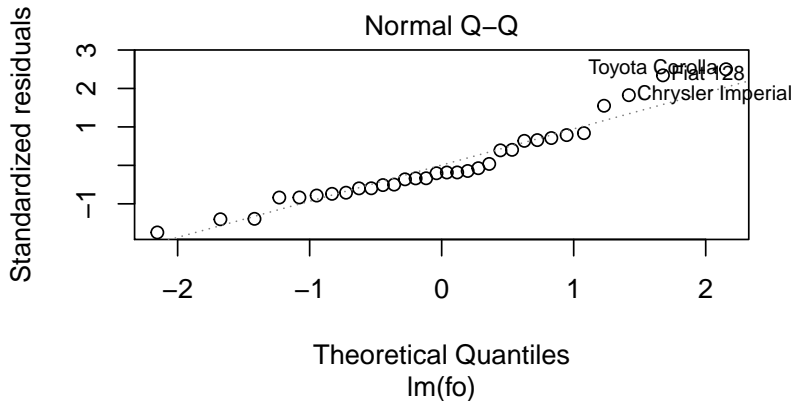
Another example for object orientation

- ▶ m3 is now a special regression object
- ▶ Various functions can be applied to this object

```
predict(m3) # Prediction
resid(m3) # Residuals
```

```
##         Mazda RX4     Mazda RX4 Wag       Datsun 710       Horn
##         22.27914          21.46545         26.25203
## Hornet Sportabout          Valiant
##         16.64696          19.59873

##         Mazda RX4     Mazda RX4 Wag       Datsun 710       Horn
##         -1.2791447        -0.4654468      -3.4520262
## Hornet Sportabout          Valiant
##         2.0530424         -1.4987281
```

Make model prediction

```
pre <- predict(m1)
head(mtcars$mpg)

## [1] 21.0 21.0 22.8 21.4 18.7 18.1

head(pre)

##           Mazda RX4      Mazda RX4 Wag        Datsun 710      Horn
##            23.28261           21.91977          24.88595
## Hornet Sportabout            Valiant
##            18.90014           18.79325
```

Regression diagnostic with base-R

Visualizing residuals

```
plot(mtcars$wt,mtcars$mpg)
abline(m1)
segments(mtcars$wt, mtcars$mpg, mtcars$wt, pre, col="red")
```
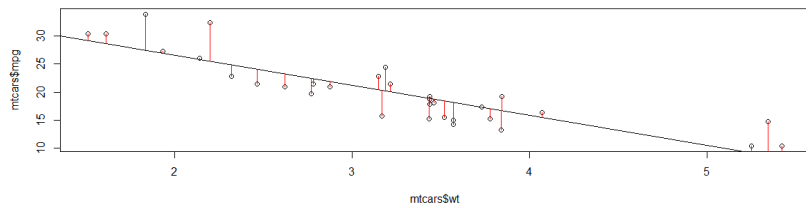
The mean squared error (mse)

- ▶ The **MSE** measures the average of the squares of the errors
- ▶ **The lower the better**

```
(mse5 <- mean((mtcars$mpg - pre)^2)) # model 5
```

```
## [1] 8.697561
```

```
(mse3 <- mean((mtcars$mpg - predict(m3))^2))
```

```
## [1] 5.974124
```

Package Metrics to compute mse

```
library(Metrics)
mse(mtcars$mpg,predict(m3))
```

```
## [1] 5.974124
```

THE VISREG-PACKAGE

```
install.packages("visreg")
library(visreg)
```
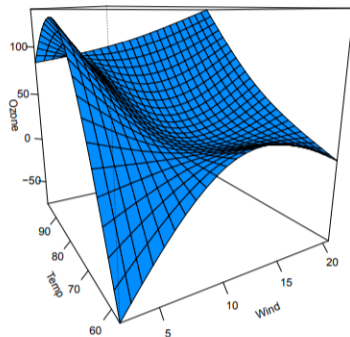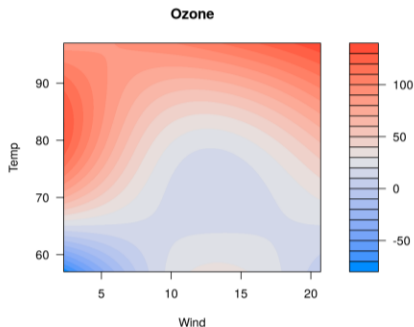
THE VISREG-PACKAGE

- ► The default-argument for `type` is `conditional`.
- ► Scatterplot of `mpg` and `wt` plus regression line and confidence bands

```
visreg(m1, "wt", type = "conditional")
```
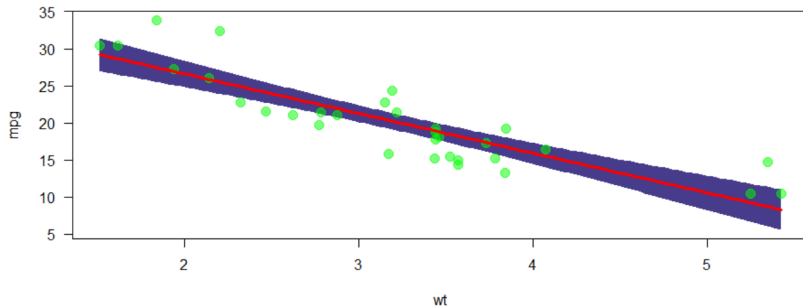
REGRESSION WITH FACTORS

▶ The effects of factors can also be visualized with visreg:

```
mtcars$cyl <- as.factor(mtcars$cyl)
m4 <- lm(mpg ~ cyl + wt, data = mtcars)
# summary(m4)

##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 33.990794  1.8877934 18.005569 6.257246e-17
## cyl6        -4.255582  1.3860728 -3.070244 4.717834e-03
## cyl8        -6.070860  1.6522878 -3.674214 9.991893e-04
## wt          -3.205613  0.7538957 -4.252065 2.130435e-04
```

Effects of factors

```
par(mfrow=c(1,2))
visreg(m4, "cyl", type = "contrast")
visreg(m4, "cyl", type = "conditional")
```
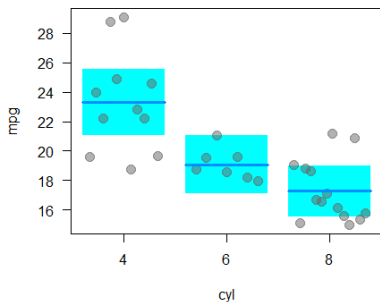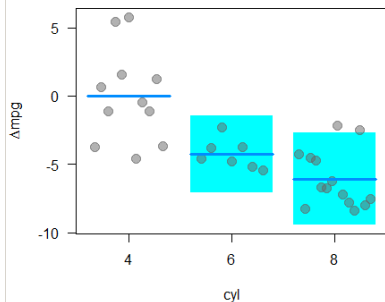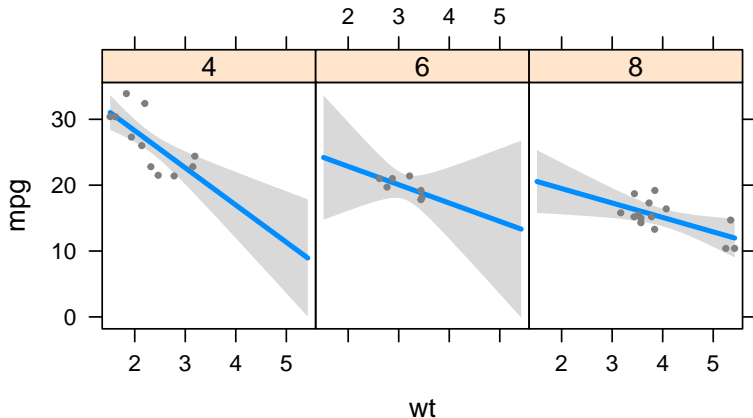
The package visreg - Interactions

```
m5 <- lm(mpg ~ cyl*wt, data = mtcars)
# summary(m5)
##                 Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  39.571196    3.193940 12.3894599 2.058359e-12
## cyl6        -11.162351    9.355346 -1.1931522 2.435843e-01
## cyl8        -15.703167    4.839464 -3.2448150 3.223216e-03
## wt           -5.647025    1.359498 -4.1537586 3.127578e-04
## cyl6:wt       2.866919    3.117330  0.9196716 3.661987e-01
## cyl8:wt       3.454587    1.627261  2.1229458 4.344037e-02
```
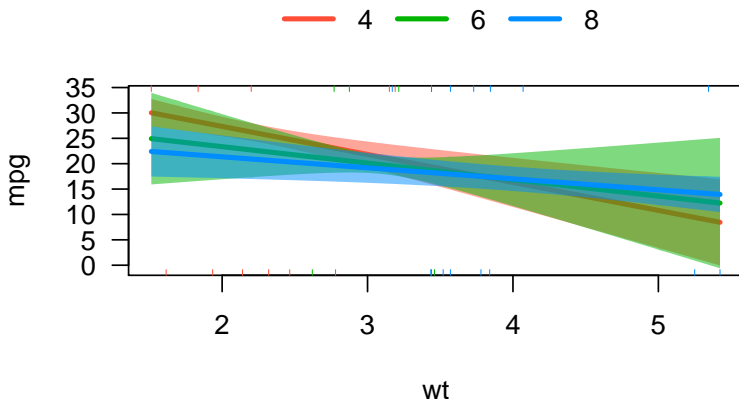
Control of the graphic output with layout.

```
visreg(m5, "wt", by = "cyl",layout=c(3,1))
```
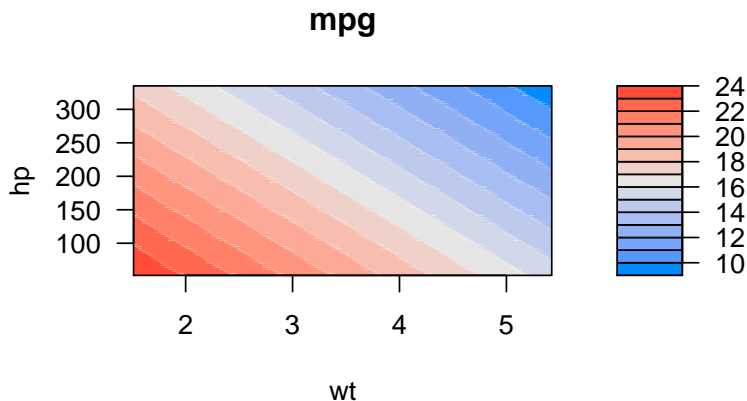
THE PACKAGE VISREG - INTERACTIONS OVERLAY

```
m6 <- lm(mpg ~ hp + wt * cyl, data = mtcars)

visreg(m6, "wt", by="cyl", overlay=TRUE, partial=FALSE)
```

```
visreg2d(m6, "wt", "hp", plot.type = "image")
```
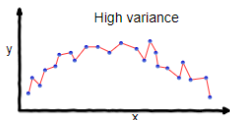
mpg

wt
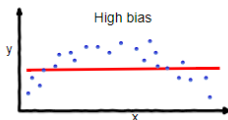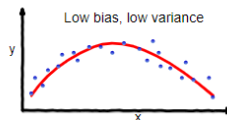
▶ The bias–variance tradeoff is the property of a set of predictive models whereby models with a lower bias in parameter estimation have a higher variance of the parameter estimates across samples, and vice versa.
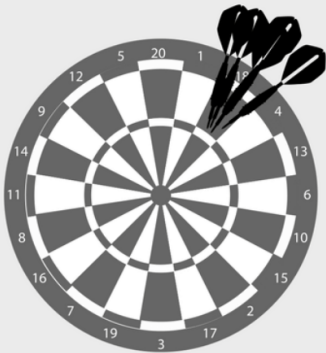

overfitting   underfitting   Good balance

THE BIAS-VARIANCE TRADEOFF (II)

High Bias
Low Variance

High Variance
Low Bias

High bias, low variance
algorithms train models that

High variance, low bias
algorithms train models that

EXERCISE: REGRESSION AMES HOUSING DATA

1) Install the package AmesHousing and create a **processed version** of the Ames housing data with (at least) the variables Sale_Price, Gr_Liv_Area and TotRms_AbvGrd
2) Create a regression model with Sale_Price as dependent and Gr_Liv_Area and TotRms_AbvGrd as independent variables. Then create seperated models for the two independent variables. Compare the results. What do you think?

The Ames Iowa Housing Data

```
ames_data <- AmesHousing::make_ames()
```

Some Variables

- ▶ Gr_Liv_Area: Above grade (ground) living area square feet
- ▶ TotRms_AbvGrd: Total rooms above grade (does not include bathrooms
- ▶ MS_SubClass: Identifies the type of dwelling involved in the sale.
- ▶ MS_Zoning: Identifies the general zoning classification of the sale.
- ▶ Lot_Frontage: Linear feet of street connected to property
- ▶ Lot_Area: Lot size in square feet
- ▶ Street: Type of road access to property
- ▶ Alley: Type of alley access to property
- ▶ Lot_Shape: General shape of property
- ▶ Land_Contour: Flatness of the propert

MULTICOLLINEARITY

▶ As p increases we are more likely to capture multiple features that have some multicollinearity.
▶ When multicollinearity exists, we often see high variability in our coefficient terms.
▶ E.g. we have a correlation of 0.801 between Gr_Liv_Area and TotRms_AbvGrd
▶ Both variables are strongly correlated to the response variable (Sale_Price).

```
ames_data <- AmesHousing::make_ames()
cor(ames_data[,c("Sale_Price","Gr_Liv_Area","TotRms_AbvGrd")])
```

```
##               Sale_Price Gr_Liv_Area TotRms_AbvGrd
## Sale_Price     1.0000000   0.7067799     0.4954744
## Gr_Liv_Area    0.7067799   1.0000000     0.8077721
## TotRms_AbvGrd  0.4954744   0.8077721     1.0000000
```

Multicollinearity

```r
lm(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd, data = ames_data)
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd, data =
##
## Coefficients:
##   (Intercept)    Gr_Liv_Area  TotRms_AbvGrd
##       42767.6          139.4       -11025.9
```

▶ When we fit a model with both these variables we get a positive coefficient for Gr_Liv_Area but a negative coefficient for TotRms_AbvGrd, suggesting one has a positive impact to Sale_Price and the other a negative impact.

Seperated models

- If we refit the model with each variable independently, they both show a positive impact.
- The Gr_Liv_Area effect is now smaller and the TotRms_AbvGrd is positive with a much larger magnitude.

```
lm(Sale_Price ~ Gr_Liv_Area, data = ames_data)$coefficients
```

```
## (Intercept) Gr_Liv_Area
##   13289.634     111.694
```

```
lm(Sale_Price ~ TotRms_AbvGrd, data = ames_data)$coefficients
```

```
##   (Intercept) TotRms_AbvGrd
##      18665.40      25163.83
```

- This is a common result when collinearity exists.
- Coefficients for correlated features become over-inflated and can fluctuate significantly.

Consequences

- One consequence of these large fluctuations in the coefficient terms is **overfitting**, which means we have high variance in the bias-variance tradeoff space.
- We can use tools such as **variance inflaction factors** (Myers, 1994) to identify and remove those strongly correlated variables, but it is not always clear which variable(s) to remove.
- Nor do we always wish to remove variables as this may be removing signal in our data.

▶ Our model doesn't generalize well from our training data to unseen data.
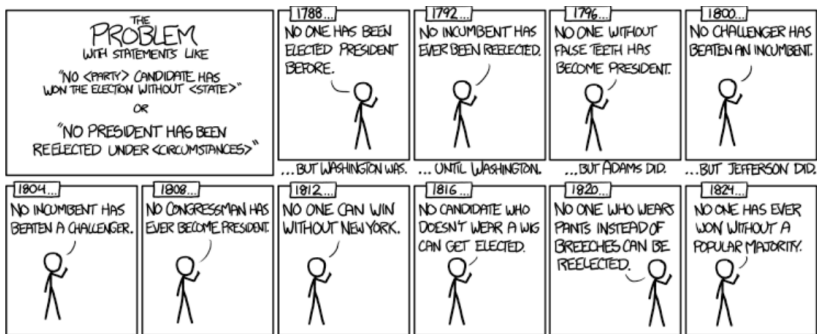
What can be done against overvitting

- **Cross Validation**
- Train with more data
- Remove features
- Regularization - e.g. ridge and lasso regression
- Ensembling - e.g. bagging and boosting

CROSS VALIDATION

- ► Cross-validation is a powerful preventative measure against overfitting.
- ► Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model.

NECESSARY PACKAGES

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

Cross Validation in R

Split data into training and testing dataset

```r
training.samples <- ames_data$Sale_Price %>%
createDataPartition(p = 0.8, list = FALSE)
train.data  <- ames_data[training.samples, ]
test.data <- ames_data[-training.samples, ]
```

Build the model and make predictions

```r
model <- lm(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd,
            data = train.data)
# Make predictions and compute the R2, RMSE and MAE
(predictions <- model %>% predict(test.data))
```

```
##         1          2          3          4          5          6
## 166826.03 126703.47 121356.31 175112.21 124232.16 148540.80 1
##         8          9         10         11         12         13
## 258106.82 225606.76 155714.61 228514.19 163855.42 114201.52 2
##        15         16         17         18         19         20
```

MODEL WITH CROSS VALIDATION

▶ Loocv: **leave one out cross validation**

```
train.control <- caret::trainControl(method = "LOOCV")

# Train the model
model2 <- train(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd,
                data = train.data, method = "lm",
                trControl = train.control)
model2 %>% predict(test.data)
```

```
##         1          2          3          4          5          6
## 166826.03 126703.47 121356.31 175112.21 124232.16 148540.80 1
##         8          9         10         11         12         13
## 258106.82 225606.76 155714.61 228514.19 163855.42 114201.52 2
##        15         16         17         18         19         20
## 110712.60 279425.83 167584.48 174499.14 205140.95 146442.42 3
##        22         23         24         25         26         27
## 158217.52 140305.23 144729.55 171124.00 171768.67 198049.33 3
##        29         30         31         32         33         34
```

Summarize the results

```
print(model2)

## Linear Regression
##
## 2346 samples
##    2 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 2345, 2345, 2345, 2345, 2345, 2345,
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   55618.48  0.5324399  38009.18
##
## Tuning parameter 'intercept' was held constant at a value of
```

Links - linear regression

- ▶ Regression - **r-bloggers**
- ▶ The complete book of **Faraway**- very intuitive
- ▶ Good introduction on **Quick-R**
- ▶ **Multiple regression**
- ▶ **15 Types of Regression you should know**
- ▶ `ggeffects` - **Create Tidy Data Frames of Marginal Effects for 'ggplot' from Model Outputs**
- ▶ **Machine learning iteration**

NICE TABLE OUTPUT WITH STARGAZER

```
library(stargazer)
stargazer(m3, type="html")
```

EXAMPLE HTML OUTPUT:

	Dependent variable:
	mpg
wt	-3.125***
	(0.911)
cyl	-1.510***
	(0.422)
am	0.176
	(1.304)
Constant	39.418***
	(2.641)
Observations	32

Shiny App - Diagnostics for linear regression

- ▶ Shiny App - **Simple Linear Regression**
- ▶ Shiny App - **Multicollinearity in multiple regression**