Introduction to Machine Learning

Module 1: Foundations

Tobias Rebholz

University of Tübingen

Fall 2024, SMiP Workshop

Orga

Some Personal Information

- SMiP alum:
  - April 2023: PhD thesis on statistical modeling in judgment and decision-making (JDM)
- Since July 2023: Postdoc in the Social Cognition and Decision Sciences group in Tübingen
  - I.e., where I also did my PhD
- From April 2025: Visiting research scholar at Duke University
  - Walter-Benjamin grant on advice taking from generative AI
- Contact: tr.rebholz@gmail.com

My Research

- 2019/20: Master's thesis on developing and benchmarking various machine learning (ML) approaches for modeling (i.e., predicting; more details later) ordinal-scaled dependent variables (= "targets" in ML jargon)
  - This is also where most of my (likely outdated!) "expertise" in this area comes from
- My current ML research covers only a very specific and limited domain
  - Often not directly related to methodological aspects
  - Instead, mainly from an augmented JDM perspective (i.e., focus on human-computer interaction)
- How I still ended up giving this workshop on the broad topic of ML in general?
  - Being excited about all kinds of methodological research (like you)
- So beware:
  A) We will mostly stay on a rather conceptual and introductory level
  B) It is very likely that some of my slides will contain errors—both small and large
  - Please let me know if you find any inconsistencies or mistakes!

Workshop Contents

- Fundamentals of machine learning (ML), such as:
  - Cross validation and hyperparameter tuning
  - Interpretable ML
- Various ML methods, such as:
  - Classification and regression trees (i.e., “Supervised Learning”)
  - Cluster analysis (i.e., “Unsupervised Learning”)
  - Neural networks (i.e., “Semi-Supervised Learning”)
  - Large language models (i.e., “Natural Language Processing”)
- Applications in various applied psych and behavioral econ areas:
  - Judgment and decision-making (JDM)
  - Management and consumer psychology
- Selected case studies, such as:
  - Analyzing the effects of social and informational influences on human JDM
  - Identifying sentiment or emotion in language
  - Predicting consumer behavior or clustering brand perceptions

Learning Targets

- Develop a basic understanding of ML and its value to applied psych/econ research
  - Emphasis is on conceptual understanding, rather than (technical) details!
- Gain practical experience in applying specific ML techniques to solve real-world data-driven decision-making problems
  - Incl. new skills: Analyze and interpret complex data sets, such as unstructured text data
- Acquire the competence to critically reflect on the results of ML methods
  - Incl. evaluating their implications for theory building and (psychological) research practice
- Prerequisites:
  - Basic understanding of statistics (e.g., linear regression)
  - Familiarity with R

Structure

- The workshop is divided into five modules:
  1. Introduction
  2. Supervised Learning
  3. Unsupervised Learning
  4. Deep Learning
  5. Interpretability (incl. Ethics)

- ... each consisting of basically two parts:
  1. Introduction to topics/methods
     - Mainly lectured by me!
  2. Solving simple programming assignments
     - Mainly hands-on programming by you!

Tentative Schedule

Thursday		Friday	
10:00 - 11:30	Module 1	09:00 - 10:30	Module 3 (cont'd)
Coffee Break (15 min)			
11:45 - 13:00	Module 2	10:45 - 12:15	Module 4
Lunch Break (1:15 h)			
14:15 - 16:00	Module 2 (cont'd)	13:30 - 15:00	Module 4 (cont'd)
Coffee Break (15 min)			
16:15 - 17:15	Module 2 (cont'd)	15:15 - 16:45	Module 5
17:15 - 18:00	Module 3	16:45 - 17:00	Wrap-Up

Material

- Slides, tutorials etc. will be provided via my personal website: https://tobiasrebholz.github.io/teaching/2024-fall-ML-SMiP
  - Password: **smip24**

Literature

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer US. https://doi.org/10.1007/978-1-0716-1418-1
   - More technical, less related to behavioral science, but more standard
   - Available for free at: https://www.statlearning.com/
   - Now also available for Python: https://doi.org/10.1007/978-3-031-38747-0
2. Jacobucci, R., Grimm, K. J., & Zhang, Z. (2023). *Machine Learning for social and behavioral research* (2nd ed.). The Guilford Press.
   - Less technical, more related to behavioral science, but less standard
   - Should be freely available for universities, at least in BW: E.g., via Proquest
3. Hastie, T. J., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed). Springer.
   - Extremely technical, but basically the ML bible

Questions

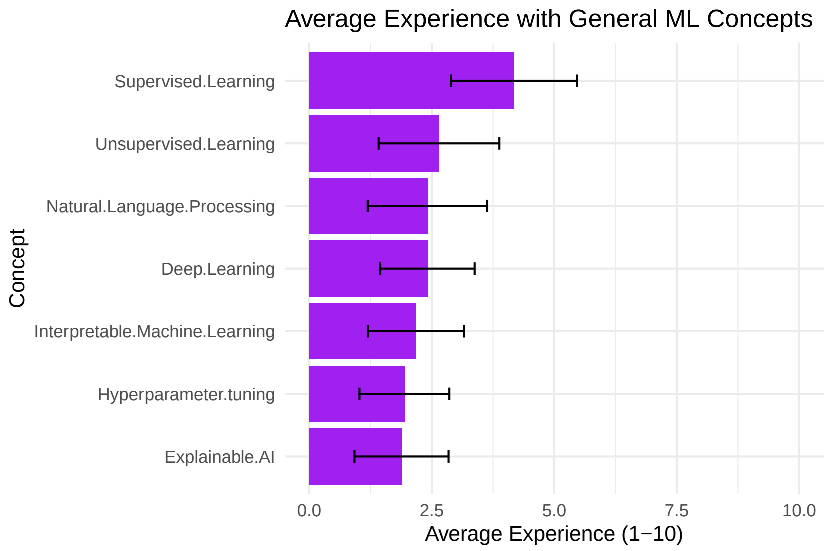
- Any open questions about the organizational details?

Module 1: Foundations

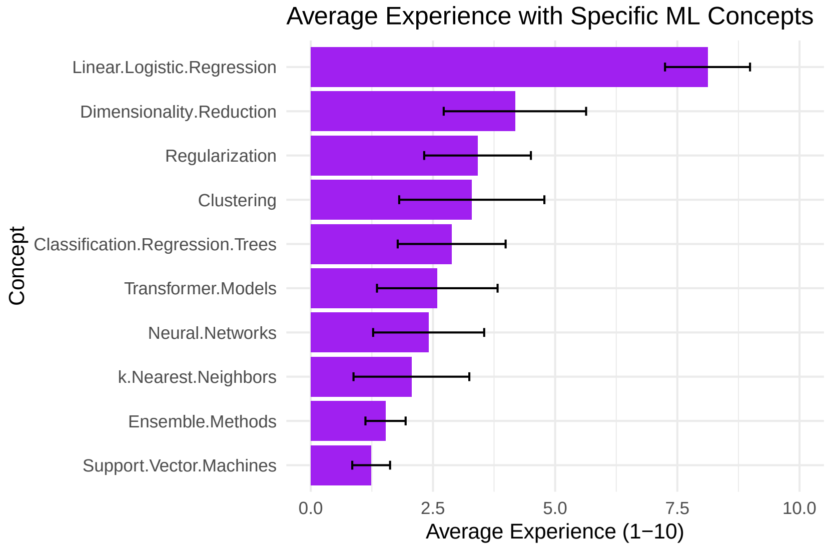
A Brief History of ML

- Early 19th century: Method of least squares was developed
  - Precursor to linear regression (see below)
- 1940s: Introduction of logistic regression for predicting qualitative values (see below)
- Early 1970s: Coining of the term “generalized linear models,” encompassing linear and logistic regression (not directly discussed!)
- 1980s: Improvement in computing technology $\Rightarrow$ Nonlinear methods
  - Mid-1980s: Development of classification/regression trees (see Module 2)
  - 1980s: Early neural networks (e.g., perceptron; see Module 4)
  - 1990s: Emergence of support vector machines (see Module 2)
  - 2001: Random Forests (i.e., ensample methods; see Module 2)
- 2010s: Deep Learning (i.e., advanced neural networks; see Module 4)
  - 2017: Transformer models (e.g., large language models; see Module 4)
    - Enabled high-performance chatbots, such as OpenAI’s ChatGPT

Your experiences



Your experiences



Analytics

- Supervised Learning: Predicting/estimating an output based on one or more inputs
- Unsupervised Learning: Learning relationships/structure from inputs only
  - I.e., there is no supervising output
- Visualization: Visual representations of predictions and relationships, but also of raw data

Important Topics

- Overfitting
- Imbalance
- Sparseness
- Robustness
- Hyperparameters

Data Types

- Continuous data
  - E.g., age, income, ...
  - Can be discretized (e.g., age categories)
- Binary data
  - E.g., yes vs. no response
  - Usually coded as a 0-1 dummy variable
- Categorical data
  - E.g., gender, group membership, ...
  - Sometimes converted into dummies: One dummy variable per category
- Ordinal data
  - E.g., highest educational degree
  - Must be treated with caution, as they are neither continuous nor categorical

Complex Data: Multivariate

- Data on demographics and Big Five personality trait scores:

```
head(dat, 15)
##       CASE gender education      age agree conscientious extra neuro open
## 652  63116      1         3 40.55747   5.8          3.40   3.6   1.5 3.80
## 999  63967      1         4 35.37340   4.6          3.60   4.0   1.0 3.60
## 991  63955      2         4 21.55923   3.0          3.20   4.0   3.8 4.40
## 392  62547      2         1 22.80091   4.8          5.20   4.4   2.0 4.80
## 788  63493      2         2 23.07484   5.2          3.40   4.4   2.6 4.60
## 330  62419      2         3 30.08120   5.4          4.00   4.4   4.0 3.80
## 2231 66716      2         5 37.61331   3.8          4.80   4.4   4.2 5.00
## 1128 64275      1         4 37.76547   5.0          5.00   4.6   3.6 4.60
## 1061 64101      1         3 17.58908   4.6          4.00   4.6   1.8 3.60
## 1474 65080      2         4 22.49090   2.8          3.80   4.0   3.0 3.00
## 1949 66088      1         2 58.15496   5.0          4.75   5.0   4.8 5.00
## 1005 63983      1         3 23.31878   3.6          4.60   3.2   2.6 3.40
## 261  62266      1         3 30.89218   3.0          3.40   3.6   4.0 4.00
## 2024 66264      2         3 24.58253   4.8          3.40   4.2   2.0 4.75
## 2441 67257      2         5 18.87679   4.8          4.40   4.0   4.6 4.00
```

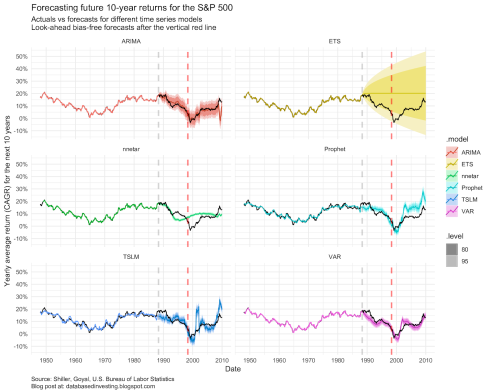
Complex Data: Spatial

- Traffic data:

(https://unece.org/traffic-census-map)

Complex Data: Time-stamped

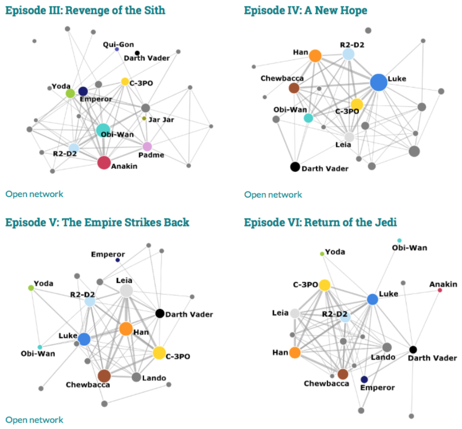
- Stock market data:



(https://www.r-bloggers.com/2020/05/forecasting-the-next-decade-in-the-stock-market-using-time-series-models/)

Complex Data: Relational

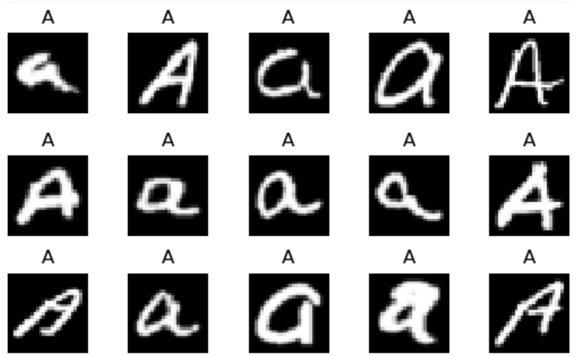
- Network data:



(https://www.martingrandjean.ch/star-wars-data-visualization/)

Complex Data: Unstructured

- Image data:



(https://github.com/zandreika/letters-recognition)

Complex Data: Unstructured

- Further unstructured data:
  - Audios
  - Videos (i.e., image + audio)
  - Text (see Module 4)
  - Health records
  - . . .

Important Statistics of Data

- Mean: $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
- Variance: $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$
- Standard deviation: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$
- Minimum: $\min = \text{minimize}_{i\in N}\{x_i\}$
- Maximum: $\max = \text{maximize}_{i\in N}\{x_i\}$

- Bringing variables to the same range (very useful for multivariate data):
  - Standarizing:
$$\frac{x-\mu}{\sigma} \tag{1}$$
  - Normalizing:
$$\frac{x-\min}{\max-\min} \tag{2}$$

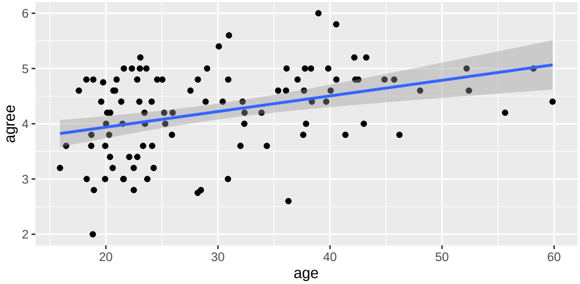
Summary

- Data Science builds mathematical models to extract and represent knowledge
- Data-driven decision-making can be relevant to any setting in:
  - Psychology
  - Economics
  - Medicine
  - Physics
  - ...
- Prior to building the models, data often needs some preprocessing

Linear Regression

Linear Regression

```r
library(tidyverse)

ggplot(dat, aes(y = agree, x = age)) +
  geom_point() +
  stat_smooth(method = "lm")
## `geom_smooth()` using formula = 'y ~ x'
```



Research Goal

- **Description:** Research with the aim of describing relationships or distributions

  vs.

- **Explanation:** Research with the aim of understanding the underlying mechanisms

  vs.

- **Prediction:** Research with the aim of maximally explaining the variability in an outcome
  - ML, and therefore this course, is mainly devoted to predictive analytics

Predictive Analytics

- There is a dependent variable or “target”, y
- There is a set of p explanatory variables or “features”, $x_1, x_2, \ldots, x_p$
- We build a model that predicts y using the information in $X = (x_1, x_2, \ldots, x_p)$, often denoted as $f(X)$
- The model differs depending on the nature of the target
  - Regression task: The target is continuous
  - Classification task: The target is discrete

Simple Linear Regression

- For each instance i in a population, we have:
  - **One** feature, x_i
  - Continuous target, $y_i \in \mathbb{R}$
- Goal: Predict the target for new instances of which we know the value of the feature, but not the value of the target:

$$x_{new} \rightarrow \hat{y}_{new} \in \mathbb{R} \tag{3}$$

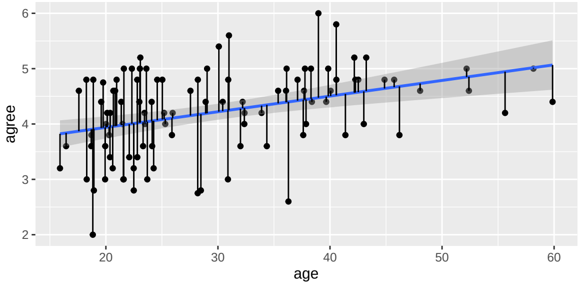
Simple Linear Regression

- We assume that there is a linear relationship between the (single) feature and the target: $y = \beta_0 + \beta_1 x + \varepsilon$
- As β_0 and β_1 are unknown, we have to estimate them from the data
  - Goal: Ensuring low errors (“residuals”), $e_i = y_i - \hat{y}_i$
    - where $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ – the target value predicted by the model
  - Note: The residuals are assumed to be normally distributed, $\varepsilon \sim N\left(0, \sigma_\varepsilon^2\right)$
- In other words, we want to find the values of β_0 and β_1 that minimize the difference between the predicted and observed target values for the entire sample
  - E.g., mean squared (prediction) error/loss function:

$$MSE = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4}$$

Simple Linear Regression

```
mdl <- lm(agree ~ age, data = dat)

mdl %>%
  ggplot(aes(y = agree, x = age)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_segment(aes(xend = age, yend = .fitted))
## `geom_smooth()` using formula = 'y ~ x'
```



Simple Linear Regression in base R

```
summary(mdl)
##
## Call:
## lm(formula = agree ~ age, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9057 -0.5915  0.1043  0.5542  1.5251
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.373290   0.223798  15.073  < 2e-16 ***
## age         0.028270   0.007092   3.986 0.000129 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7274 on 98 degrees of freedom
## Multiple R-squared:  0.1395, Adjusted R-squared:  0.1307
## F-statistic: 15.89 on 1 and 98 DF,  p-value: 0.0001292
```

- Fitted model: $ag\hat{r}ee = \hat{\beta}_0 + \hat{\beta}_1 * age = 3.37 + 0.03 * age$
  - E.g., prediction for a new participant of $age = 30$: $ag\hat{r}ee = 3.37 + 0.03 * 30 = 4.22$

Prediction “Out-of-Sample”

1. Use a subset of the sample to fit the model:

```
N <- nrow(dat)
train <- sample(1:N, N*0.9)

df_subset <- dat[train,]

mdl_subset <- lm(agree ~ age, data = df_subset)
summary(mdl_subset)
##
## Call:
## lm(formula = agree ~ age, data = df_subset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9365 -0.6068  0.1003  0.5975  1.5335
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.440593   0.246566  13.954  < 2e-16 ***
## age         0.026329   0.007975   3.301  0.00139 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7449 on 88 degrees of freedom
## Multiple R-squared:  0.1102, Adjusted R-squared:  0.1001
```

Prediction “Out-of-Sample”

2. Test the model on the remaining, held-out “test” data:

```
df_rest <- dat[-train,]

df_rest$agree_predicted <- predict(mdl_subset, df_rest)

df_rest %>% select(agree, agree_predicted)
##      agree agree_predicted
## 1949   5.0        4.971744
## 274    2.8        3.939006
## 2378   4.8        4.560227
## 1672   5.0        4.204977
## 1000   4.2        3.970284
## 737    4.8        4.644516
## 1754   4.8        4.621869
## 2550   4.0        4.106741
## 816    3.2        4.079394
## 744    4.0        4.437448
```

- We will expand on this idea over the next modules

Simple Linear Regression in `mlr3`

1. Define a (prediction) task, which is mlr3's way to store the raw data along with some meta-information for modeling
   - E.g., specify what kind of task to be completed
2. Specify which ML model to apply later for prediction
   - `mlr3` does not implement its own ML models but links to available implementations in other R packages
     - E.g., "regr.lm" links to the ordinary `lm()` function in the stats package
3. Train the linear regression model
   - In `mlr3`, objects have "abilities" (or "methods") that can be applied with the following $-syntax (cf. other object-oriented programming languages, such as Python)
     - Here, the train method of the learner object is used to fit the model ("train the learner") from step 2 on the task specified in step 1:

```r
library(mlr3verse)
tsk = as_task_regr(agree ~ age, data = dat) #1.
mdl = lrn("regr.lm") #2.
mdl$train(tsk) #3.
```

Simple Linear Regression in `mlr3`

- The estimated model output is exactly the same as for base R
  - Not surprising, as `lrn("regr.lm")` is simply applying the `lm()` function to estimate the model, which is exactly equivalent to what we did a couple of slides ago

```
summary(mdl$model)
##
## Call:
## stats::lm(formula = task$formula(), data = task$data())
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9057 -0.5915  0.1043  0.5542  1.5251
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.373290   0.223798  15.073  < 2e-16 ***
## age         0.028270   0.007092   3.986 0.000129 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7274 on 98 degrees of freedom
## Multiple R-squared:  0.1395, Adjusted R-squared:  0.1307
## F-statistic: 15.89 on 1 and 98 DF,  p-value: 0.0001292
```

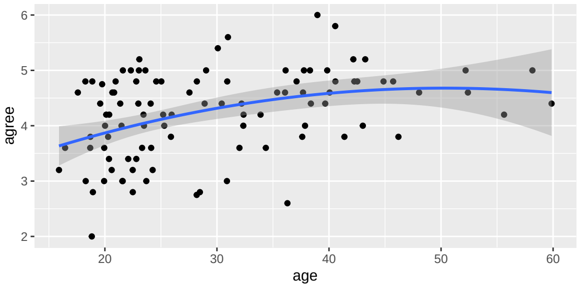
Excurse: Why `mlr3`?

- If the end result (i.e., fitted model) is exactly the same as with `base` R, why should we bother about using `mlr3` in the first place?
- In `mlr3`, the model fitting procedure (i.e., steps 1-3) is exactly the same **no matter what ML method we're using** to model the data
  - In other words, it (merely) provides a **unified framework/syntax** (cf. `tidyverse`) to fit different ML models to data
    - The ML models themselves, however, stem from other, established R packages (e.g., "regr.lm" links to the ordinary `lm()` function in the stats package), which are also described in detail in, e.g., James et al. (2021)
  - Allows us to **focus more on the concepts** and less on the specific programming in R
- `mlr3` is **the state-of-the-art framework for ML in R** (cf. `scikit-learn` in Python)

Excurse: Nonlinear Regression

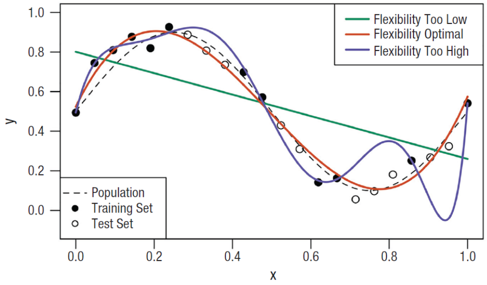
- Using linear modeling, we can even build nonlinear models
  - E.g., by including polynomial transformations of features (i.e., quadratic age): $a g\hat{r} e e = \hat{\beta}_0 + \hat{\beta}_1 * age + \hat{\beta}_2 * age^2$

```
ggplot(dat, aes(y = agree, x = age)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ x + I(x^2))
```



Bias-Variance Trade-Off

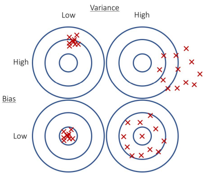
- Bias-variance trade-off: Good out-of-sample performance requires low variance as well as low squared bias
  - Why "trade-off"? – "Inflexible" model with low variance but high bias *vs.* "flexible" model with low bias but high variance
  - The challenge lies in finding a model for which both the variance and the (squared) bias are low



(Pargent et al., 2023, Figure 3a)

Bias-Variance Trade-Off

- Darts metaphor:
  - Bullseye = True value (i.e., expected target value for a given combination of feature values)
  - Each cross is the prediction made by a concrete ML model
    - Models are trained on a randomly drawn training set, all from the same population and with the same sample size
- Optimal model: Low bias and low variance (bottom left)
  - Noise = Irreducible error of the true model
    - Reason why predictions (across different samples) are not similar, even when hitting the bullseye



(Pargent et al., 2023, ESM, Figure 2)

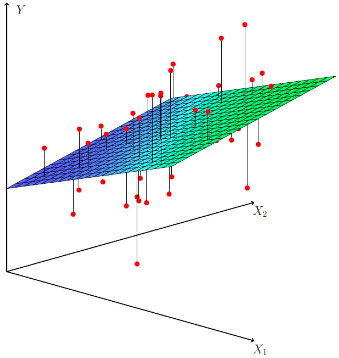
Multiple Linear Regression

- Everything is the same as in simple regression
  - Except that we now have multiple features (incl. polynomial transformations of features to build nonlinear models)
  - Note: Higher dimensionality (i.e., more features) = More flexibility
- For each instance i in a population, we have:
  - A **vector** of features, $X_i = (x_{i1}, x_{i2}, \dots, x_{ip})$
  - Continuous target, $y_i \in \mathbb{R}$
- Goal: Predict the target for new instances of which we know the vector of features but not the value of the target:

$$X_{new} \to \hat{y}_{new} \in \mathbb{R} \tag{5}$$

Multiple Linear Regression

- E.g., vector of features of size 2:



(James et al., 2021, Figure 3.4)

Multiple Linear Regression in base R

```
mdl <- lm(agree ~ age + gender, data = dat)
summary(mdl)
##
## Call:
## lm(formula = agree ~ age + gender, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.86767 -0.49583  0.07832  0.53912  1.45493
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.040273   0.351015   8.661 1.04e-13 ***
## age         0.028927   0.007093   4.078 9.31e-05 ***
## gender      0.188803   0.153585   1.229    0.222
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7255 on 97 degrees of freedom
## Multiple R-squared:  0.1527, Adjusted R-squared:  0.1353
## F-statistic: 8.743 on 2 and 97 DF,  p-value: 0.0003229
```

Multiple Linear Regression in `mlr3`

```r
tsk = as_task_regr(agree ~ age + gender, data = dat)
mdl = lrn("regr.lm")
mdl$train(tsk)
summary(mdl$model)
##
## Call:
## stats::lm(formula = task$formula(), data = task$data())
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.86767 -0.49583  0.07832  0.53912  1.45493
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.040273   0.351015   8.661 1.04e-13 ***
## age         0.028927   0.007093   4.078 9.31e-05 ***
## gender      0.188803   0.153585   1.229    0.222
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7255 on 97 degrees of freedom
## Multiple R-squared:  0.1527, Adjusted R-squared:  0.1353
## F-statistic: 8.743 on 2 and 97 DF,  p-value: 0.0003229
```

Hands-on Practical Tutorial

- Now it’s your turn:
  - Go to https://tobiasrebholz.github.io/teaching
  - Select “Introduction to Machine Learning” > “Materials”
    - Password: **smip24**
  - Download the “Linear Regression” tutorial
  - Work through the tasks

Logistic Regression

Logistic Regression

- Everything is the same as in linear regression
  - Except that we now have a discrete target
- For each instance i in a population, we have:
  - A vector of features, $X_i = (x_{i1}, x_{i2}, \dots, x_{ip})$
  - **Binary** class membership, $y_i \in \{0, 1\}$
    - E.g., buying vs. not buying a specific product
- Goal: Predict the target (i.e., class membership) for new instances of which we know the vector of features but not the value of the target:

$$X_{new} \rightarrow \hat{y}_{new} \in \mathbb{R} \tag{6}$$

Logistic Regression

- Auxiliary target: Probability of membership in class 1, p
  - Counter probability $1 - p$: Membership in class 0
  - **Continuous, but bounded** target
- Auxiliary goal: Predict the auxiliary target (i.e., probability) for new instances of which we know the vector of features but not the value of the target:

$$X_{new} \rightarrow \hat{p}_{new} \in (0, 1) \tag{7}$$

- Predicted class:

$$\hat{y}_{new} = \begin{cases} 1 \text{ if } \hat{p}_{new} > 0.5 \\ 0 \text{ else} \end{cases} \tag{8}$$

Logistic Regression

- Fitting a logistic function to the probability of class 1:

$$p = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}} \tag{9}$$

  - Is equivalent to fitting a linear function to the logarithm of the odds:

$$\log\left(\frac{p}{1-p}\right) = \beta_0+\beta_1 x_1+\cdots+\beta_p x_p \tag{10}$$

- In general, "odds" is defined as the probability of an event occurring relative to the probability of the event not occurring
  - Here, the odds are the probability for membership in class 1, relative to the probability for membership in class 0
- As $\beta_0, \beta_1, \ldots, \beta_p$ are unknown, we have to estimate them from the data
  - E.g., by maximizing the log likelihood function $\Rightarrow \hat{\beta}_p$ is called the "maximum likelihood estimate" (MLE)

Logistic Regression in mlr3

- Data preparation:

```
dat$agree_high <- ifelse(dat$agree > 4, 1, 0)
dat %>% select(agree, agree_high) %>% tail(., 10)
##      agree agree_high
## 442    4.8          1
## 2270   5.0          1
## 1133   5.2          1
## 1210   4.8          1
## 1912   4.6          1
## 744    4.0          0
## 1485   5.0          1
## 9      4.0          0
## 866    3.6          0
## 1591   4.4          1
```

Logistic Regression in mlr3

- Model fitting:

```
tsk = as_task_classif(agree_high ~ age, data = dat, positive = "1")
mdl = lrn("classif.log_reg")
mdl$train(tsk)
summary(mdl$model)
##
## Call:
## stats::glm(formula = task$formula(), family = "binomial", data = data,
##     model = FALSE)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.83118    0.73681  -2.485  0.01294 *
## age          0.07940    0.02561   3.100  0.00193 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 133.75  on 99  degrees of freedom
## Residual deviance: 121.83  on 98  degrees of freedom
## AIC: 125.83
##
## Number of Fisher Scoring iterations: 4
```

Logistic Regression in mlr3

- Fitted model: $\hat{p}_{agree>4} = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 * age}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 * age}} = \frac{e^{-1.83+0.08*age}}{1+e^{-1.83+0.08*age}}$
    - Prediction for a new participant with $age = 30$:
      $\hat{p}_{agree>4} = \frac{e^{-1.83+0.08*30}}{1+e^{-1.83+0.08*30}} = 0.63$

```
summary(mdl$model)$coefficients
##                Estimate Std. Error   z value    Pr(>|z|)
## (Intercept) -1.83118304 0.73680914 -2.485288 0.012944661
## age          0.07940429 0.02561037  3.100474 0.001932111
```

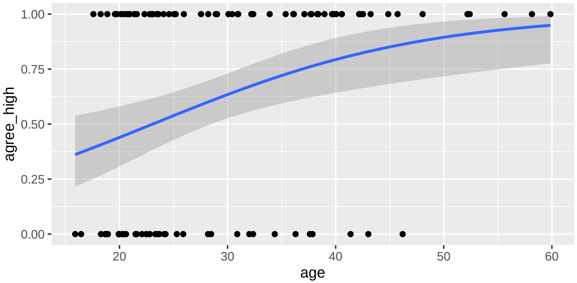
- Odds ratios:

```
exp(coefficients(mdl$model))
## (Intercept)         age
##   0.1602239   1.0826419
```

Logistic Regression in `mlr3`

- We can plot the estimated probability of class 1 membership as a function of age:

```
ggplot(dat, aes(y = agree_high, x = age)) +
  geom_point() +
  geom_smooth(method = "glm", method.args = list(family = binomial(link = "logit")))
## `geom_smooth()` using formula = 'y ~ x'
```



Classification Performance

- Main goal: Correct classification
  - E.g., minimizing the mean misclassification error:

$$MMCE = \frac{1}{N}\sum_{i=1}^{N} I\{y_i \neq \hat{y}_i\} \tag{11}$$

    - Where $I\{\cdot\}$ is the indicator function taking the value 1 if the condition in the parentheses is true and 0 otherwise
  - This is equivalent to maximizing classification accuracy:

$$Acc = 1 - MMCE = \frac{N_{0,0} + N_{1,1}}{N} \tag{12}$$

- Confusion matrix:

	$\hat{y} = 0$	$\hat{y} = 1$	
$y = 0$	$N_{0,0}$ (TN)	$N_{0,1}$ (FP)	$N_{0,\cdot}$
$y = 1$	$N_{1,0}$ (FN)	$N_{1,1}$ (TP)	$N_{1,\cdot}$
	$N_{\cdot,0}$	$N_{\cdot,1}$	N

Classification Performance

- Other important metrics:
  - Sensitivity (or true positive rate, recall):

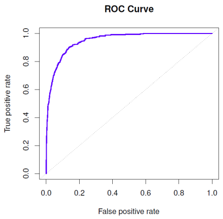
$$Sens = \frac{TP}{TP + FN} \tag{13}$$

  - Specificity (or true negative rate):

$$Spec = \frac{TN}{TN + FP} \tag{14}$$

Classification Performance

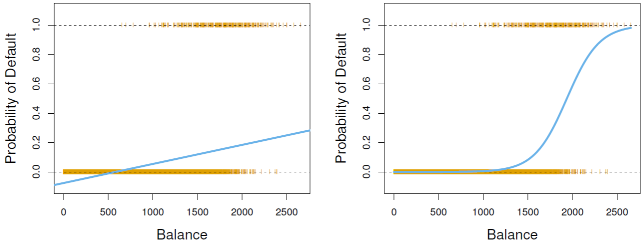
- Predicted class: $\hat{y} = \begin{cases} 1 \text{ if } \hat{p} > 0.5 \\ 0 \text{ else} \end{cases}$
  - Note: Thresholds other than 0.5 may lead to better performance
- Area under the (receiver operating) curve (AUC):
  - The probability that an observation randomly drawn from class 1 has a higher predicted probability to belong to class 1 than an observation randomly drawn from class 0
  - The ROC traces out $Sens$ and $Spec$ for varying classification thresholds:



(James et al., 2021, Figure 4.8)

Excurse: Logistic vs. Linear Regression

- Linear regression: Some estimated probabilities are negative
  - I.e., ill-defined for binary target
- Logistic regression: All estimated probabilities lie between 0 and 1
  - I.e., well-defined for binary target



(James et al., 2021, Figure 4.2)

Hands-on Practical Tutorial

- Now it's your turn:
  - Go to https://tobiasrebholz.github.io/teaching
  - Select "Introduction to Machine Learning" > "Materials"
    - Password: **smip24**
  - Download the "Logistic Regression" tutorial
  - Work through the tasks

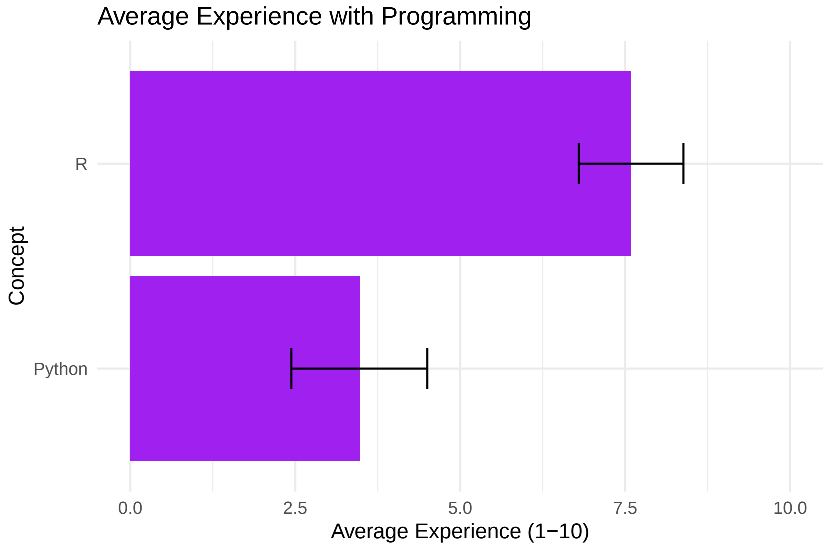
Summary

Summary

- Machine Learning (ML) is as easy and straightforward as:
  - Linear regression: Building models to estimate the (linear) relationship between a continuous target variable and a set of features
    - Goal: Predict target values of new instances
  - Logistic regression: Building models to estimate the probability of (binary) class membership based on a set of features
    - Goal: Predict class memberships of new instances
- Before building the model, the data may need to be preprocessed
  - E.g., normalized or standardized features, transformed targets, ...
- To expand:
  - Other, more advanced (supervised and unsupervised) learning algorithms
  - Variable selection
  - Cross validation

Appendix: R for Beginners

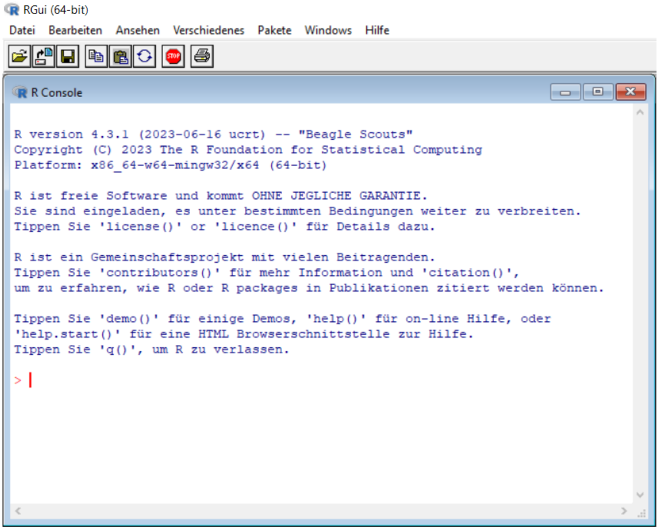
Your experiences



Quick Facts about R

- It is a programming language
- It contains tools from Statistics, Data Mining, Machine Learning, Visualization, . . .
- It has good graphical facilities
- It includes cutting-edge technology
- It is Open Source
- It runs on different platforms (Windows, MacOS, UNIX)
- It has extensive documentation for help

R Console



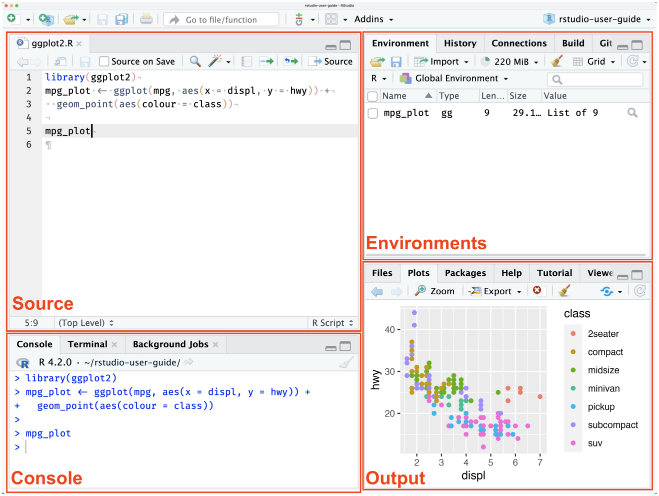
```
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R ist freie Software und kommt OHNE JEGLICHE GARANTIE.
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.
Tippen Sie 'license()' or 'licence()' für Details dazu.

R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.
Tippen Sie 'contributors()' für mehr Information und 'citation()',
um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.

Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.

> |
```

Installation

- Go to http://www.r-project.org
- Click on the download link on the main page
- Choose your preferred CRAN mirror
- Navigate to “Download and Install R”
- Choose your platform

- Important: We will be working with R Version 4.3.3
  - Same version makes troubleshooting much easier
  - You can check your installed version by executing `R.Version()` in your console

RStudio Editor



```
library(ggplot2)
mpg_plot <- ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(colour = class))

mpg_plot
```

```
5:9    (Top Level)    R Script
```

Name	Type	Len...	Size	Value
mpg_plot	gg	9	29.1…	List of 9

```
R 4.2.0 · ~/rstudio-user-guide/
> library(ggplot2)
> mpg_plot <- ggplot(mpg, aes(x = displ, y = hwy)) +
+   geom_point(aes(colour = class))
>
> mpg_plot
>
```

Installation

- Before starting, make sure that R is installed correctly
- Go to https://posit.co/downloads/
- Click on the download link on the main page
- Navigate to "Install RStudio"
- Choose your platform

Coding in R

- Command lines start with ">"
- Comment lines start with "#"
- Use "<-" or "=" to assign a value to a variable
- Use "==", "<", ">", and "!=" to check logical conditions

Variables, Vectors, Matrices, . . .

- Variables: Define a variable, `x`, assign it the value 1, print `x`, add 5 to `x`, print `x` again

```
x <- 1
print(x)
## [1] 1

x <- x + 5
print(x)
## [1] 6
```

- Vectors: Define two vectors, `u = (1, 4, 10, -1)` and `v = (10, -4, 3, 0)`, print `u + v`

```
u <- c(1, 4, 10, -1)
v <- c(10, -4, 3, 0)

z = u + v
print(z)
## [1] 11  0 13 -1
```

Variables, Vectors, Matrices, . . .

- Matrices: Define a 4 × 2 matrix with columns u and v, print the matrix

```
mtrx <- cbind(u, v)
print(mtrx)
##       u  v
## [1,]  1 10
## [2,]  4 -4
## [3,] 10  3
## [4,] -1  0
```

- Data frame: Closely related to the concept of a matrix
  - The rows represent individual observations or “instances” (lines 1-4) and the columns represent variables (u and v)

```
as.data.frame(mtrx)
##    u  v
## 1  1 10
## 2  4 -4
## 3 10  3
## 4 -1  0
```

Arithmetics

- Mathematical operations:

```
x <- 10
y <- 7

x + y
## [1] 17

x - y
## [1] 3

x * y
## [1] 70

x / y
## [1] 1.428571

x %% y
## [1] 3
```

Functions

- Mathematical functions:

```
x
## [1] 10

sqrt(x)
## [1] 3.162278

v
## [1] 10 -4  3  0

abs(v)
## [1] 10  4  3  0
```

- Logical functions:

```
if (x >= 1) {
  print(TRUE)
} else {
  print(FALSE)
}
## [1] TRUE
```

Functions

- Statistical functions:

```
u
## [1]  1  4 10 -1

mean(u)
## [1] 3.5

sd(u)
## [1] 4.795832

median(u)
## [1] 2.5

summary(u)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -1.0     0.5     2.5     3.5     5.5    10.0
```

Dataset Handling

- Reading from/writing to a file:

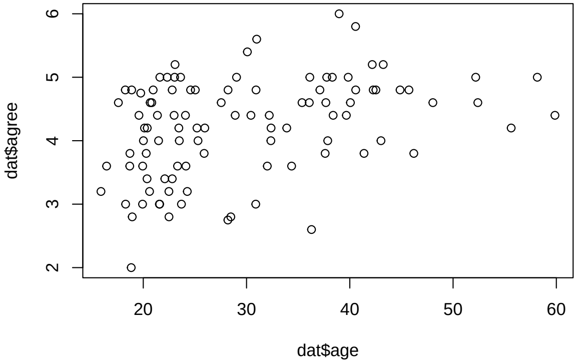
```
#reading data from drive
dat <- read.csv('subfiles/data/bfi.csv', header = TRUE)
head(dat)
##    CASE gender education      age agree conscientious extra neuro open
## 1 63116      1         3 40.55747   5.8           3.4   3.6   1.5  3.8
## 2 63967      1         4 35.37340   4.6           3.6   4.0   1.0  3.6
## 3 63955      2         4 21.55923   3.0           3.2   4.0   3.8  4.4
## 4 62547      2         1 22.80091   4.8           5.2   4.4   2.0  4.8
## 5 63493      2         2 23.07484   5.2           3.4   4.4   2.6  4.6
## 6 62419      2         3 30.08120   5.4           4.0   4.4   4.0  3.8

# writing data to drive
write.csv(dat, file = 'subfiles/data/bfi.csv', row.names = FALSE)
```

- Well-known data repositories for research purposes:
  - http://archive.ics.uci.edu/ml/
  - https://www.kaggle.com/datasets

Plotting

- Scatterplot:

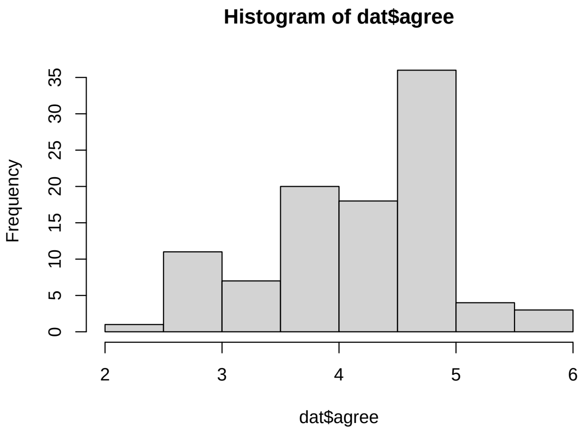
```
plot(dat$agree ~ dat$age)
```



Plotting

- Histogram:

```r
hist(dat$agree)
```



Packages

- Some packages are included by default, such as `stats` and `graphics`
  - Use `installed.packages()` to find out which ones
- Other packages can be installed using `install.packages("name")`
  - Packages provide extended functionality for R
    - Most important package for data handling: `tidyverse`
    - Most important package for ML: `mlr3verse`
  - Ideally, install packages with `dependencies = TRUE` to also include any packages that might be used in the package `name` you are installing
- You have to load new packages with `library("name")` every time you start a new session
  - Alternatively, you can access individual functions from installed packages without loading the entire package (cf. Python) with `name::function()`

Packages

- Caution: Different packages can have the same names for different functions
  - The last loaded package overwrites any functions with the same name in previously loaded packages
- Recommendations:
  - Load all packages needed for the analysis in one section at the beginning of the script
  - Usually it is best to load `tidyverse` and `mlr3verse` last
  - Pay attention to warnings when loading packages
    - If certain functions are overwritten, you can still access them with `name::function()`
  - Always start a new R session when you start your analysis to avoid unexpected behavior from packages loaded from previous sessions

More to Come

- You can access the internal help with `?command`
- Google (and recently also GPT) are your best friends for solving programming issues
- With the practical tutorials, we will get more familiar with R