Cognitive Modeling (INFOMCM)
General lab information

- Work in groups of 2. Sign your group up via Blackboard.
- Work in either R or Python (your choice). Quick introductions are online.
- Work on your own device. We recommend both students work parallel on their own computer, so you both know what's going on.
- **Hand in for grading:**
    - Only answers to the questions labelled "Blackboard questions"
    - Upload those via Blackboard (link in folder of assignment)
    - Use the dedicated answer sheet (in upload link)
- Teachers are only available during scheduled class hours.
- This year ('22- '23) we use NEW assignments. Any "teething problems", we keep track of in a "rolling document" on Blackboard. Your input is welcome.
- Throughout the course you work on three bigger assignments:
    1. Process models: you learn how theory informs models (3 points)
    2. Machine learning models: you learn how data informs models (3 pnts)
    3. Bayesian models: you learn how to compare models (4 points)
- Grading:
    - Your lab grade is the sum of the individual labs. It is expected that not every student will get the full points on each lab.
    - Some labs have bonus questions for extra points.
    - Points are typically awarded for insight; not for code. Answer the question at hand, don't "dump" code or system output (unless asked).
- See course manual for further details on these aspects:
    - Deadlines are strictly enforced
    - Plagiarism will not be tolerated
    - Health comes first: if you are sick, please stay at home. No need to e-mail the teachers. Have a student in class to connect with the teachers and ask questions. We are running around a lot & can't check e-mail or MS Teams.
- Have fun! We let you learn by challenging you.

Symbols in this assignment

Blackboard question: This is a question that will be graded. The answer needs to be handed in via the dedicated Blackboard upload link. This can be a PDF.

Engineering: These steps that are needed to complete the assignment, and typically involve coding. For example, to make a modification to the model. The code itself and substeps are needed, but not graded (unless specified).

Self-check and hints: These help you check whether you understood the material. They are not graded, we encourage you to write down the answers for yourself.

Testing: You can do these questions to check if your intuition is correct.

Lab Assignment 1: Processing models

Developer and coordinator: Chris Janssen
Teachers: see course manual

Background and goals

Different modeling techniques have different uses. *Processing models* are useful in cases where there is some *theory or idea* about the relevant cognitive *process*, from which you want to derive *predictions* for a novel situation. For example, to predict the outcome of an experiment, or to predict how humans uses different interfaces and to choose what design best fits your application.

At the end of this lab session, you can:

1. Implement components of a processing model (at the cognitive band / algorithmic level).
2. Make predictions for different use cases using a processing model that is informed by theory.
3. Evaluate the outcomes of a processing model to make a recommendation.

Structure and time-management

You work on the topic of driver distraction in three subsections:

1. Section 1 introduces the basic components of a processing model
2. Section 2 introduces a model of texting-while-driving, that will be extended to make predictions for writing e-mails while driving.
3. Section 3 then refines this model to look at alternative strategies for performing the task.

Below is a suggested time-line for the average student. If you are faster: try the bonus assignment. If you are slower: catch up after class hours. There are two sessions where staff is present:

- Session 1: Finish section 1 and start section 2 (Blackboard questions 1 and 2). For example, finish the "word" strategy model.
- Session 2: Finish section 2 and 3 (blackboard questions 1 and 2). Optional: bonus exercises.

If you are relatively new to programming:

- Make use of the teaching staff and check with them whether you are "on track" *regularly*. We are here to help.
- Pick the programming language you are most familiar with. If you are in doubt between R and Python, we recommend R.
- Make sure you have gone through the quick introduction / refresher of your programming language before the first lab.
- Clear your calendar: you might need to do more work outside of class.
- Come prepared to each lab session with questions.

Section 1: Basic concepts

1.a. Core: the model human (information) processor

At their core, many *cognitive processing models* describe how humans process information. Stuart Card, Tom Moran, and Allen Newell laid the foundation in their 1983 book "The psychology of human-computer interaction". They were among the first to use *theoretical* insights from psychology to *predict* how humans use computers.

The predictions were made in code, but were so clear that they could often be captured in simple images. The following is one example for a simple reaction task. It is described in more detail on the next page.

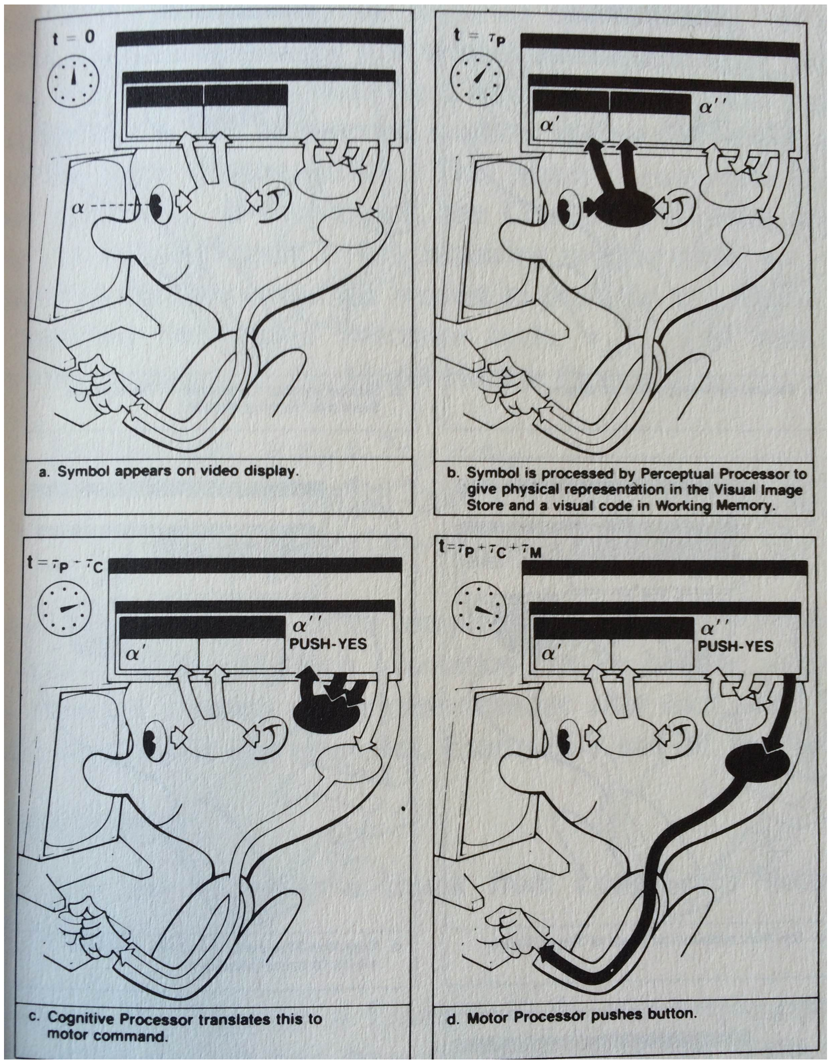
t = 0
α
a. Symbol appears on video display.
t = τP
α'
α''
b. Symbol is processed by Perceptual Processor to give physical representation in the Visual Image Store and a visual code in Working Memory.
t = τP + τC
α'
α''
PUSH-YES
c. Cognitive Processor translates this to motor command.
t = τP + τC + τM
α'
α''
PUSH-YES
d. Motor Processor pushes button.

Figure 1: Example of model human processor task (from Card, Moran, Newell, 1983, page 67)

Definition of human task: The human is asked to press a button as soon as they see a symbol on the screen.

Card, Moran, and Newell (1983) systematically go through 3 steps:
Step 1: Task analysis. This is the implementation of model/theory. The model describes *where* a cognitive process takes place (In the figure: black arrows) and *what* this process entails (in the figure: text in the black boxes). Descriptions are based on a theory (or substantial idea) of the cognitive process. Typically, one task has many subtasks, or subcomponents that need to be specified. You want to get to a level where each subtask can be attributed to specific cognitive processes.
Figure 1 has 4 steps (panel a,b,c,d):

A. At time t = 0, the stimulus appears (symbol α on computer screen). Nothing has happened in the brain yet (no black arrows, no filled black boxes).
B. Next, a *perceptual* process takes place. There are arrows around perceptual areas of the brain/mind, and an impression of symbol α (namely: α') is stored in visual working memory, and an impression of that impression is in working memory (α'').
C. Then a *cognitive* process takes place. The human prepares the motor action that needs to be taken based on what they see (look at location of black arrows; see that "push-yes" was decided).
D. Finally, based on the action a *motor* process is executed: a button is pressed.

Step 2: Calculation: Once the core processes are known through task analysis, one can calculate *how long* each step of the process takes. In the Figure: look at the clock and the sum behind "t =". The time depends on the type of process that is involved. In this example they are nicely isolated. There are perceptual processes (time: τ_p), cognitive processes (time τ_c), and motor processes (time τ_m). In this example we focus on time calculations, but in other use cases you might also focus on other relevant calculations for the brain such as the likelihood of an error.

Step 3: Approximation: Now we approximate the unknowns, so we can use the model to make a prediction. We want to work like (physical) engineers who use approximations all the time. For example, when determining "can a bridge hold the weight of all cars on it", an engineer approximates how many cars can fit on the bridge and their weight. Is this below the weight level that the bridge can take? Of course, there are many details that can be considered. But the engineer only needs to consider approximations of all RELEVANT aspects for their level of abstraction.

This is the same in cognitive processing models: there are many detailed processes in the brain, but we want to *approximate* the core processes that are relevant to the model at the *level of abstraction* that we need. For a good processing model, we want theory to inform the decision. Using outcomes of other experiments, Card, Moran, and Newell set the times of the different processes at:
$\tau_p = 100$ ms
$\tau_c = 70$ ms
$\tau_m = 70$ ms

<u>Engineering:</u> Implement the model that is shown in Figure 1 and described afterwards as computer code:

A. Decide whether you want to work in R or Python for this entire assignment and open your favorite editing environment.
B. Write a function for each of the processing steps and name these functions: "start", "perceptualstep", "cognitivestep", and "motorstep". The functions have no input and return as output the time this step took in milliseconds
C. Write a function "example1" that calls the functions in turn and calculates the total time. As output, the function returns this total time.

<u>Self-check:</u> If you run the function "example1" in your code, you should get the value 240 returned. (Hopefully this was incredibly simple to do – I know, you did not need code to calculate it! But we expand from here)

1.b. Accounting for variability through Fastman, Middleman and Slowman

For the earlier example of the physical engineer that estimated whether a bridge holds the weight of cars, maybe you thought *"it matters whether that vehicle is a tiny 'mini' or a big heavy truck"*. Correct! Engineers typically consider ranges of values (or scenarios). Here: a scenario where the bridge is filled with heavy trucks.

Card, Moran, and Newell also account for such variability by explicitly considering variability on each task step (perceptual, cognitive, or motor step). The process is sometimes "average", sometimes "faster" and sometimes "slower". They call this the "Middleman", "Fastman" and "Slowman" predictions. This naturally aligns with psychological experiments in which often some variability in responses is observed, and a prediction is made about the "typical" human and its variability. For psychology engineering cases (such as: "Is this interface safe to use in a car"), you also want to think about the extremes ("might an accident happen?").

Card, Moran, and Newell give the following estimates (based on *theory*) for the three types of processes (perceptual, cognitive, motor; rows) and three types of people (Fastman, Middleman, Slowman; columns):

		Fastman	**Middleman**	**Slowman**
Perceptual process	τ_p	50 ms	100 ms	200 ms
Cognitive process	τ_c	25 ms	70 ms	170 ms
Motor process	τ_m	30 ms	70 ms	100 ms

Engineering: Implement the Fastman, Middleman, Slowman concept in your example:

A. Revise each functions from example1 such that they have an input parameter "type" with as default value "middle" (for middleman).
B. Within the functions, output should now change based on whether the "type" parameter has value "fast", "middle", or "slow".
C. The code for "example1" should still work (due to use of default values).
D. Write another function called "example2" with input parameter "completeness" that has default value "extremes" and alternative value "all".
E. If the input to "example2" is "extremes", it should calculate how long the task takes for (a) a simulation that does all steps as fastman, (b) a simulation that does all steps as middleman, and (c) a simulation that does all steps as slowman. As output, it should return the three numbers in the above order.
F. If the input to the function "example2" is "all", then the function should calculate all possible combinations of processes when each process can be fastman, middleman, or slowman (i.e., 3 steps with each 3 possible values, so 3 x 3 x 3 outcomes). As output, the function should (1) print the values of each simulation, and (2) produce a box-and-whisker plot of the various outcomes.

Self-check and hints

- For question E: you should get output: 105, 240,470
- For question F:
  - With R you can make a box-and-whisker plot using an existing native function
  - Python might need additional packages. Google is your friend to find the right package for "box and whisker plot" in python ;-)
- Below is what my box-and-whisker plot looks like in R. Note that I added a *label* for the y-axis, and that 0 is part of the *range* on the y-axes)
- Do you know why the prediction for middle-man for question E is not aligned with the thick line of the boxplot? (hint: think about what the different lines of a boxplot mean).

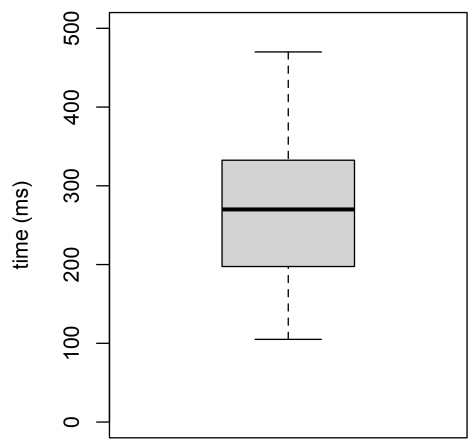

Figure 2: Example box-and-whisker plot

1.c. A more complex example

In the previous example only 1 stimulus was shown, and one output was given. Now, let's envision the following task:

Two stimuli are presented, one after the other. The human is asked to press a button if the two stimuli are identical. You will implement three versions of this task and use those models to answer the blackboard questions.

Self-check: before you read the instructions for coding this task below, think about the three steps from Card, Moran, and Newell (1983). What would be the task analysis, calculation, and approximation for this task?

Engineering: example 3

First, let's implement the code for an *ideal* example. Call this function *example3*() and go through these steps:

A. One stimulus is shown at time t= 0. The initial perceptual processing of this stimulus take t = τ_p .
B. The second stimulus is shown, again taking τ_p for perceptual.
C. Two stimuli are compared in working memory. This takes τ_c time.
D. A motor command is prepared taking τ_c .
E. The motor action is executed taking τ_m .
F. Implement a model that can make fastman, middleman, and slowman predictions similar to example 2. You can assume that if a model is slow for a specific type of step (e.g., perceptual) that they are this consistently within 1 trial or simulation, but that whether they are slow/middle/fast for different types of steps is independent (e.g., if for perception they are slow, they can be fast for motor).

This example makes the simplification that all stimuli come at the right time and do not disturb (e.g., delay) the processes.
With this model you should be able to answer Blackboard questions 1A and 1B. You can already answer them in the answer sheet (download this from Blackboard; it is attached to the submission link).

Engineering: example 4

Now let's make a model that depends on the environment: how fast the model can respond depends on how fast the stimulus is presented. Call this function example4.

A. Start by copying code from example3
B. Your function needs a *for-loop* that runs the model for different timings of the second stimulus. The timing of the second stimulus is relative to time t=0 (trial start) and has options: 40, 80, 110, 150, 210, and 240 ms.

C. (For simplicity's sake) make the assumption that the first stimulus is processed normally without interference or dependence on presentation of the second stimuluas (as you know from class; this is not always the case, such as in PRP studies)
D. However, the second stimulus will only start processing *after* the perceptual process of the first stimulus has finished. At that time, the perceptual processing will take place and again take the necessary processing time.
E. Implement a model that can make fastman, middleman, and slowman predictions similar to example 2 and example 3.
F. Again, assume that if the model is slow for a specific type of step (e.g., perceptual) that this is consistent within 1 trial or simulation, but that whether the model is slow/middle/fast for different types of steps is independent (e.g., perception, cognitive, motor).

You should observe how the timing of your stimuli impacts the time predictions. You can now answer Blackboard question 1C.

Engineering: example 5

In experiments, people often make more errors when they are fast. These relationships are very delicate and involve complex processes. Yet, we can make a fastman, slowman, middleman prediction to approximate the effects. Take the code from example 3 and adjust it such that it calculates both a time prediction AND an error prediction, as follows:

A. The model starts with a basic error probability of 0.01 (or 1%)
B. The total error likelihood is now a multiplication of factors
    i. For each middleman step, the error probability is doubled (i.e., x 2)
    ii. For each slowman step, the error probability is halved (i.e., x 0.5)
    iii. For each fastman step, the error probability is tripled (i.e., x 3)
C. Calculate for all possible combinations of fastman, middleman, slowman processes (3 step types x 3 speeds) what the trial time and expected error probability is.
D. Make a scatterplot that shows on the horizontal axis trial time and on the vertical axis error likelihood (either in a percentage, or as a fraction)

This can be used to answer Blackboard question 1D.

Blackboard question 1 (1 point total)

Answer the following Blackboard questions in the dedicated answer sheet (see upload link):

A. For example 3: what is the predicted task time for the model that does all steps as fastman in milliseconds?
B. For example 3: What is the predicted task time for a model that has the fastest perception, takes average time for cognitive steps, but is the slowest in motor execution?
C. For example 4: What is the predicted slowest time that we might observe in this experiment?

D. For example 5: add a picture or screenshot of the scatterplot (as an image), Make sure to clearly label the x- and y-axis (give name and measurement unit, e.g. “time (ms)”) and to use an appropriate range of values on each axis.

1.d. Summary of part 1

Cognitive processing models are simulations that make useful predictions. Similar to how physical engineers make predictions of the environment, you:

1. Describe the task / situation (**task analysis**). We make the assumption that people indeed do the task in the best way they can (‘rationality principle’).
2. Make **calculations**, for which you can use theory as input. This is different from a typical statistical test in an experiment where the outcome is binary (significant difference yes/no; effect yes/no). Here we work towards scenarios (how large is an effect, under which conditions?).
3. Know that the world is not always exactly as the mean/average and use **approximations** to capture variability. We looked at the fastman, middleman, slowman principle. There are other ways possible of course, such as thinking about serial vs parallel processes.

Section 2: Applied scenario: e-mailing while driving

You will now apply these principles of using models and theory about cognitive processes to make predictions in an example: sending e-mails while driving.

2.a. Starting point: summary of Janssen's driver distraction model when typing digits

We want to model how distracting it is to write e-mails while driving. Computational cognitive process models are useful here, as this scenario would not be ethical to test on the road. As a starting point we use an existing driver distraction model of Janssen and colleagues (e.g., Janssen & Brumby 2010; Janssen, Brumby, Garnett, 2012). The fundamental principles of that model are also summarized here: https://www.youtube.com/watch?v=rQhj0vjVZFU

The following is the essence that is needed for the course:

- The original model models a task where drivers alternate between steering and typing in the digits of a phone number
- When drivers use their mobile phone (to type in digits), this takes their eyes and hands (and thereby: minds) away from the road/traffic.
- When drivers use the phone, they are not controlling the car, and the car starts drifting away from the center lane of the road (a common metric of driver distraction). This drift can be captured in mathematical equations.
- When drivers return attention to the road, they use the steering wheel to adjust the position of the car to get closer to the center of the car. The angle at which the steering wheel is held depends on the position of the car relative to lane center (the further away you are, the more strongly the adjustment). This can be captured in mathematical equations.
- When people switch between driving and dialing and vice-versa, there is a switch cost: this takes some time.

Similar to the examples in section 1, the model goes through stages. However, now these stages are bigger *tasks.* The model is either "driving", "dialing", or "switching" between the two. That is, the model is specified at a higher level of abstraction.

The model can be used to systematically explore what the impact is of how many digits are dialed between segments of driving: 1, 2, 3, 4, or 5 digits. This impacts the duration of the distraction and therefore the safety of the car, but also the time needed to call someone. Both can be calculated systematically in the model.

2.B. New research question: writing e-mails while driving

The studies by Janssen and colleagues used a task that is hardly done anymore: manually typing phone numbers on a phone with explicit digits. However, people still perform distracting tasks in the car. Some even write e-mails while driving. How distracting is that? And does it matter how attention is divided between writing e-mails and steering the car? You will adjust the Janssen model to make predictions for such scenarios.

To do this, these models and theories are provided:

1. A model of how the car drifts when the driver is not steering. This follows a cumulative random normal function with *M*= 0 and *SD* =0.13 (for details: see Janssen & Brumby 2010 and work cited there)
2. A model of how the position of the car is updated using lateral velocity (latVel), and as a function of lane deviation (LD):
    a. latVel = 0.2617 * LD^2 + 0.0233 * LD – 0.022
    b. Intuition: large values of LD have substantially larger effects on latVel (exponentially)
    c. The car's latVel is adjusted using a physical wheel that is limited in its angle. Therefore, latVel is limited between -1.7 m/s and + 1.7 m/s
    d. In driver studies the car is not always "perfect" at lane center, but at some distance. The car starts at 0.27 m (cf. Janssen et al).
    e. Driving updates are done in steps of 250 ms (cf. Salvucci, 2005).
3. A model of how attention is interleaved between the two (cf. Janssen et al.):
    a. We assume attention is paid to one task at a time (serial, not parallel)
    b. A switch is needed to go from one task to the other
4. Theory of 2 finger typing on mobile phone:
    a. *M*=39.33 words per minute (*SD*=10.3WPM) (Jiang et al, 2020; table 1)
    b. The number of words varies per sentence. The internet suggested 15-20 word / sentence (e.g., https://techcomm.nz/Story?Action=View&Story_id=106)
    c. An e-mail consist of multiple sentences and multiple words.
5. Interleaving and thinking:
    a. When think of the next sentence, this takes time which you (in part) do not spend on the road. I guestimate this at 300 ms.
    b. When you switch from driving to typing after each word, then coming up with the next word takes some time ("where was I again?") that is (in part) not spent on the road. I guestimate this at 200 ms.
6. An idea of how people might divide their attention between tasks; the different *strategies* for performing the task. The literature suggest that people tend to interleave at "natural breakpoints" (see Janssen, Brumby, Garnett, 2012 for review): points in the task structure where some aspects/subtasks have been completed and it is more "natural" to switch then at others. When we extrapolate to e-mailing while driving, this could be:
    a. **NOT** typing at all (just driving)
    b. Doing some driving after every **WORD**
    c. Doing some driving after every **SENTENCE**
    d. Doing some driving after every **EMAIL**

With this model it is interesting to see how the four different strategies impact performance. How much worse is driver safety? And how much faster/slower is an e-mail typed? You will explore this with your own model!

Self-check

With every model and experiment, it is useful to hypothesize early what you think the outcome might be and why (ideally, cf. theory). For the four strategies above:

- How safe do *you* think this strategy is? (i.e., how much will the car deviate?)
- How fast will the e-mailing task be?
- What critical factors / parameters might impact this behavior?
- Beyond a qualitative pattern (or relative ordering): what order of magnitude do you expect that these effects will have? For example: How much more will the car deviate in one condition compared to the other? How much slower might the driver be in writing the e-mail?

With these hypotheses identified for yourself, you can later check if you were thinking the right direction. And… the model will have quantified your thoughts!

Open the driver model

On Blackboard you can download the basic code to make a driver distraction model. Depending on your programming language: "drivermodel.R" or "drivermodel.py". Read through the code top to bottom and see if you understand what each section and function does.

2.C. A model of interleaving after every WORD

Create a model that interleaves typing for driving after every word (needed for Blackboard question 2A)

Within the function "runTrial" you need to write code for different strategies. Start with modeling 1 strategy: interleaving after every word. Follow these steps:

1. Call the function resetParameters().
2. Make two variables:
   a. locDrifts: a vector that stores all the calculated drift values. You need these to plot how the car deviates over time (see later in assignment)
   b. trialTime: stores current trial time (number)
3. Make an if-else-loop to checks if the strategy on this trial is "word".
4. Initiate the trialtime to 0, and the vehicle position to the start position.
5. Throughout the code, with each step that the model takes you want to update the trialtime and the position of the vehicle
6. Calculate the typing speed of the current simulated trial, by sampling from a normal distribution using the known parameters of WPM. This will be the speed of the current simulation trial (and might change on other trials). You will need to transform the value of WPM to the time that is needed to type 1 word (in milliseconds): *timePerWord*
7. Iterate through sentences, and within a sentence iterate through words
8. Calculate how long it takes to type a word and add this to the trial time. This is a sum of:
   a. *timePerWord*
   b. *time needed to retrieve a word.*

    c. If a word is the first word of a new sentence*:* the *time needed to retrieve a sentence*

9. During this time, the vehicle will drift. This is done every 50 milliseconds.
    a. Round (using floor functions) to a rounded (integer) number of updates.
    b. For each update: update the vehicle position based on the function when the person is not steering
    c. Store these drift values in *locDrifts*
10. If you have not reached the end of the last sentence, there is a steering update:
    a. Update a rounded (integer) number of steering movements (this is a parameter that is passed to the "runTrial" function). Each such steering movements takes 250 ms (cf. Salvucci, 2005).
    b. For each of these steering movements, you calculate what the lateral velocity is of the vehicle based on the current lateral position
    c. This velocity is in meters / second. However, every 50 msec the car's drift should be calculated. So, you update the drift value every 50
    d. Update the total trial time
11. Once you are done (you've iterated through the for-loops): cross check that at every step you have calculated a time and multiple lane positions.
12. You now have a value "totalTrialTime" (1 value) and a vector of lateral deviation (drift) positions. Make a scatter plot in which you show:
    a. How lane position changes over time (make the assumption that each lane position is calculated in steps of 50 ms)
        i. Show time on the x-axes in milliseconds (not steps!)
        ii. Show lane position on the vertical axis in meters
    b. Write somewhere in the plot "Mean time = XXX; Mean drift = YYY; Max drift = ZZZ". In this sentence, XXX, YYY, and ZZZ should be values that you calculate yourself.

With this model you can answer **blackboard question 2A.**

<u>Create a model that interleaves after every sentence (needed for Blackboard question 2B and 2C)</u>
Within the "runTrial" function, build another conditional that checks if the interleaving strategy equals "sentence". This model interleaves after every sentence. You can probably reuse a lot of your code from the previous assignment. However, we make the simplification that there is no retrieval time needed per word (only per sentence) – the reasoning behind this is that someone is continuing to type the sentence and does not need to think "what comes next within this sentence".
Make sure that you can again plot the outcome.
Once you have this strategy done, look at the self-check question below, and then answer the Blackboard questions 2B and 2C.

Self-check:
When you look at the plots of the model that interleave after every sentence or at every word, you should be able to pick up some patterns. Check:

- Is the total trial time approximately the same as the time at which the number of drifts end (due to some rounding these might not be exactly the same – but they should approximate)
- Doe mean and max drift correspond with what you see in the graph?
- If you compare the two different strategies: do you see where the interleaving happens?
- Do you see a difference in performance between different runs of the simulation of a similar strategy? What might cause this?

Blackboard question 2 (1 point total)
Answer the following Blackboard questions in the dedicated answer sheet (see upload link):

A. Create 10 plots and paste a picture or screenshot of them in the answer sheet. Each of plot should be a *different* outcome of running the "runTrial" function with parameters nrWordsPerSentence=17, nrSentences = 10, nrSteeringMovementsWhenSteering = 4, and interleaving="word". Please mind the criteria that were mentioned at the end of the engineering assignment for that model.
B. Create another 10 plots and paste a picture or screenshot of them in the answer sheet. These plots should be 10 different outcomes of running the "runTrial" function with the same parameter settings as for question 2A, except that the interleaving strategy should be "sentence".
C. Explain what you see in the Figures. Specifically:
   i. How does the pattern of the different strategy show itself in the Figures? (e.g., how does this differ between the file you submitted for A and the file you submitted for B)
   ii. Within one type of strategy: what causes that each individual plot is different from the others? What causes these differences?

Section 3: Comparing the interleaving strategies to baselines across many simulations

3.A. Implementing baseline strategies

To understand what the impact is of the interleaving strategy (section 2C), you will implement two baseline models to compare performance to:
1. A model that does not type at all, but only drives (interleaving = “drivingOnly”)
2. A model that only types, while the car is drifting (interleaving = “none”)

<u>Implement baseline strategies</u>
Implement each of the two baseline models mentioned above. Do this again within the if-else loop in the “runTrial” function. For the model that only drives (and does not type) you should calculate performance over a longer time interval, as if the model WOULD have typed.
So, for both of these models you calculate how long the typing would take in total (during which there is either steering or drifting), based on:
- A unique calculation of time needed per word
- Considering that retrieval of a sentence takes time (but not of each word)
- Number of words and number of sentences per word that are passed as parameters to the “runTrial” function

<u>Self-check:</u>
The code for the 2 baseline models is probably A LOT shorter. Check what a plot of deviation over time looks like for each strategy (similar to the questions of 2.C). Is this what you would expect for this strategy?

3.B. Implementing loop code

Now that the function “runTrial” can run trials for various strategies, you can implement the code for “runSimulations” (to run all simulations for multiple trials).

<u>Make code for 1 simulation faster</u>
Before, the function runTrial gave a plot as output of the simulation. However, we want to comment that code out and instead have the function return: trial time, mean drift value, and max drift value in a way that is fitting for your programming language (for example, as a vector or list).

<u>Implement function “runSimulations”</u>
This function should start by creating four vectors to store the output of each individual simulation:
1. totalTime
2. meanDeviation
3. maxDeviation

4. Condition

Then, iterate through all four interleaving conditions.
Then, iterate through the number of simulations (if your computer can take it: try 100; if that takes too much time do something like 50; else 25).
For each simulation trial (i.e., each call to runTrial):

- In a uniform way, pick a random number from the set (15, 16, 17, ..20) for nrWordsPerSentence
- set nrSentences to 10
- set nrSteeringMovementsWhenSteering to 4

For each simulation trial (runTrial) that is run, store the output in the four vectors (totalTime, meanDeviation, maxDeviation, Condition).

Make sure to test your code first with a low number of trials (e.g., 3) to see if it works in principle. If your code takes a long time, maybe you:

- kept some plotting functions on,
- have not closed a for- or if-loop properly
- are not properly resetting your parameters including internal vectors of the function.

Then use the data in the four vectors to make 1 plot that shows:

- The max lateral deviation in meters (y-axis) as a function of total trial time in milliseconds (x-axis)
- The datapoints of individual trials (i.e., per condition, per simulation 1 data point; if you ran 100 simulations per condition; you should have 400 datapoints total). Each condition should have its own unique plotting symbol (e.g., circle, triangle, cross, square). Give these points the color grey.
- The average max lateral deviation position per condition. Calculate per condition the mean of the maxDeviation and the mean of the trial times. Give these values the same symbol as you used for the raw trials of that condition (e.g., circle, triangle, cross, square). However, give each condition's mean a distinct color (e.g., blue, red, green, black), and make these symbols slightly bigger in size.
- Calculate for each condition the standard deviation for time and for lateral deviation. Plot these as error bars on top of the graph (i.e., below/above the mean; and left/right of the mean)
- Include a legend in the graph that shows which symbol and which color corresponds to which condition

You will need this plot for Blackboard question 3A, 3B and 3C. Before you answer those, make sure to check the self-check and the summary below, which make more explicit what you've learned in this assignment.

Self-check
Do the outcomes of the plot correspond with the predictions you made in section 2B? If not, do they make sense now that you know what the model does?

In a sense you've implemented a different way to consider variability in performance. Compared to the fastman, middleman, slowman technique of section 1, you do not make predictions for specific types of performers (fast/middle/slow) but rather look at how variability in strategy and trial can impact performance (e.g., variability might come from for example typing speed, number of words per sentence, and drift of the car). Consideration of such alternatives are at the basis of the computational rationality approach that was discussed in class. The fuller computational rationality approach would also use other means (such as reinforcement learning) to learn what the "best" strategy is.

3.C. Summary of parts 2 and 3

In parts 2 and 3 you have scaled the basic principles from Card, Moran, and Newell's approach to larger, more applied scenarios. In these scenarios you can also:

1. Do **a task analysis:** which components are in which model and strategy?
2. **Calculate** what impact the steps and strategies have on performance (speed of writing an e-mail, and drift of the car). We went beyond a mere "is one condition worse than the other" to a quantified situation where we can measure "how much worse".
3. **Approximate** behavior. For some aspects of the task (such as number of words per sentence, number words) we did not know the values, but we made reasonable approximations. We also knew that people can vary, and therefore ran simulations with alternative values. This can give an estimate of how likely specific outcomes are. Depending on your criterion of success, you can use this information to inform design (do you want to make a conservative estimate of safety for example?).

Of course, many more aspects of the model can be varied, and the impact of that variation can be tested. Model results could be compared to experimental results from studies. A more complete computational rationality model would now also explore what the most optimal strategy is.

Overall, even this simple model has been useful though in making predictions of behavior. You have applied an engineering mindset to understand and quantify a psychological phenomenon. Theory has guided your decisions.

If you carefully did the assignment, you should have achieved the learning goals. Now, at the end of this lab, you can:

1. Implement components of a processing model (at the cognitive band / algorithmic level) → you implemented a fastman-middleman-slowman model, and a model of driver distraction.
2. Make predictions for different use cases using a processing model that is informed by theory. → you made predictions of how performance changes in a driver distraction setting depending on interleaving strategy (and associated parameters)
3. Evaluate the outcomes of a processing model to make a recommendation → you used the outcomes of the models to make recommendations (see Blackboard question 3).

Blackboard question 3 (1 point total)
Answer the following Blackboard questions in the dedicated answer sheet (see upload link):

A. Submit a picture or screenshot of the plot that you generated (in the function "runSimulations") which shows the predictions of individual trials (horizontal axis: total trial time, vertical axis: max lateral deviation) and their mean performance + standard deviation (see above engineering question). This should be a plot for 100 simulations per condition (so 400 simulations total). If there are grounds why your computer could not do this, submit a plot based on fewer trials (e.g., 50 trials).
B. Explain using your plot from question A whether you would recommend drivers to interleave after every word or after every sentence. In your explanation make sure to refer to patterns that can be seen in the Figure, such as average performance in a condition, variability within that condition, and how the strategies impact both time for writing an e-mail and maximum deviation of the car.
C. The model could be expanded in many ways. One aspect to consider is that people do not always have their phones in their hands, but sometimes further away, such as on their lap or in a cradle. They would then have to reach for the phone before typing, and reach back to the steering wheel afterwards. Explain if and how such reaching behavior (a motor "switch cost") impacts each of the four strategies that you simulated (none, drivingOnly, word, sentence). You do not need to implement this model, but rather explain what pattern you would *expect* and *why*.

When you are done with your questions, make a PDF file from your answer sheet and have 1 group member upload it on behalf of the entire group. If you are up for a challenge, consider completing the bonus questions.

Section 4. Bonus questions (max 0.5 point; *optional*)

If you are up for an extra challenge, you can submit a bonus assignment. This is optional.
Bonus assignments should be made without any input from the teachers. They will be marked very strict. Merely handing something in will not give you points. You can get at most 0.5 bonuspoints for this lab. You can add your bonus assignment together with your other Blackboard assignments in the answer sheet.

The bonus assignment would be to implement one of the scenarios below (A or B) and then answer these four questions for the scenario you implemented:

1. What scenario did you model (interleaving after X words, or error making)
2. Submit a screenshot or PDF of the *critical part* of your code
3. Submit a plot of what performance now looks like. The plot should be in the style of Blackboard question 3, but adjusted to the scenario at hand.
4. Explain *how* and *why* performance of the model has (not) changed compared to the strategies that you modelled before.

Options for consideration:

A. A model that interleaves after every couple of words (instead of after every word, or after every sentence). There are various ways to implement this (e.g., have a fixed number of words, or a variable number of words typed each time). Argue why you chose what you chose. (Difficulty estimate: medium)

B. A model that occasionally makes a (theoretical) typing error. When a typing error is made it might be spotted immediately, or only once a word or sentence is finished. At that time, the user should do some driving and then return to correct the typo (e.g., press "backspace" until they're back at where the typo was made; make explicit in your answer how you modelled the time it takes to correct errors and why), then drive again, then continue the sentence. This can be implemented in various ways (for example, with various probabilities of errors). Argue why you chose what you chose. (Difficulty estimate: hard).

References

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Janssen, C. P. (2012). *Understanding strategic adaptation in dual-task situations as cognitively bounded rational behavior*. London, United Kingdom: UCL. PhD thesis

Janssen, C. P., & Brumby, D. P. (2010). Strategic adaptation to performance objectives in a dual-task setting. *Cognitive science*, *34*(8), 1548-1560.

Janssen, C. P., Brumby, D. P., & Garnett, R. (2012). Natural break points: The influence of priorities and cognitive and motor cues on dual-task interleaving. *Journal of Cognitive Engineering and Decision Making*, *6*(1), 5-29.

Jiang, X., Li, Y., Jokinen, J. P., Hirvola, V. B., Oulasvirta, A., & Ren, X. (2020, April). How we type: Eye and finger movement strategies in mobile typing. In *Proceedings of the 2020 CHI conference on human factors in computing systems* (pp. 1-14).

Marr, D. (1982). Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. San Francisco, CA: Freeman.

Salvucci, D. D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science*, 29, 457– 492.