NEU502B Homework 3: Drift-diffusion models

Due March 20, 2024

Submission instructions: First, rename your homework notebook to include your name (e.g. `homework-3-nastase.ipynb`); keep your homework notebook in the `homework` directory of your clone of the class repository. Prior to submitting, restart the kernel and run all cells (see *Kernel > Restart Kernel and Run All Cells...*) to make sure your code runs and the figures render properly. Only include cells with necessary code or answers; don't include extra cells used for troubleshooting. To submit, `git add`, `git commit`, and `git push` your homework to your fork of the class repository, then make a pull request on GitHub to sync your homework into the class repository.

The **drift-diffusion model** (DDM) has been a dominant model in understanding how decisions are made. The model predicts how one makes a decision by integrating evidence over time. Part of the power of the drift-diffusion model is how simple it is. The dynamics are captured by:

$$dC = v \cdot dt + w \cdot \mathcal{N}(0, 1)$$

Where C is your decision variable, v is the evidence that accumulates over time (the "drift"), and w is the amount of noise in the integration (the "diffusion"). The initial condition is typically set to C_0 = 0 and the integration occurs to some bound $\pm B$, which initiates the choice. To get a better intuition for how the model works, let's simulate a decision process using the drift-diffusion model.

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%autosave 60
sns.set(style='white', palette='colorblind')
cpal = sns.color_palette()
```

```
/tmp/ipykernel_290899/3546905863.py:2: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major r
elease of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, an
d better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issu
es/54466 (https://github.com/pandas-dev/pandas/issues/54466)

  import pandas as pd

Autosaving every 60 seconds
```

Problem 1: Simulate a simple decision process

First, we'll simulate a simple two-alternative forced choice (2AFC) decision process using the drift-diffusion model. The following function takes in evidence v and uses a `while` loop to continue updating the decision variable C until it reaches either the positive or negative decision bound $\pm B$ (with random-normal noise scaled w at each update). Make sure you understand what each line of the function is doing. We'll start with parameters $C_0 = 0$, $B = 1$, $dt = 0.0005$ ms, $v = 2.5$, and $w = 0.05$.

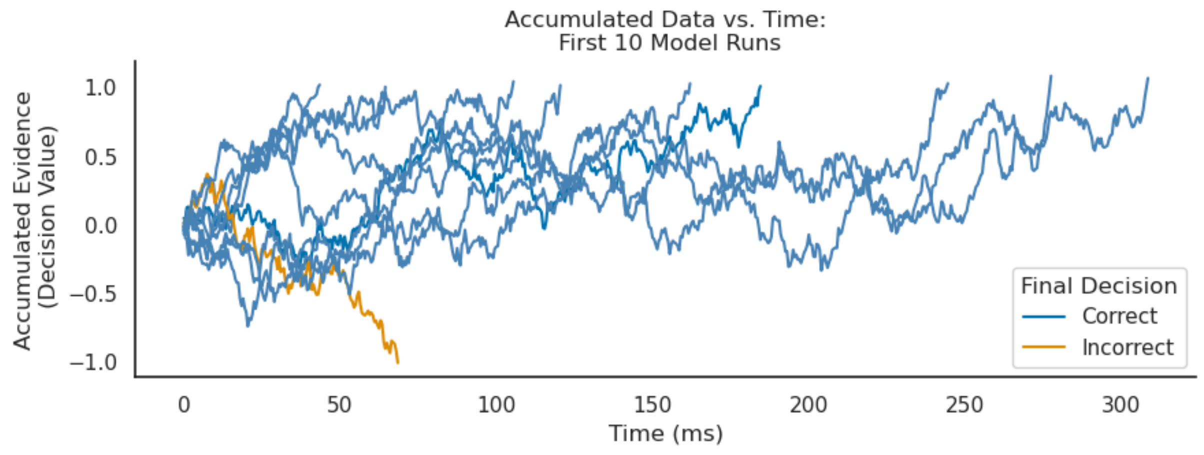
In [2]:
```
def ddm(C, v, dt, w, B):
    '''Simulate drift-diffusion model dynamics.

    Parameters
    ----------
    C : float
        Decision variable.
    v : float
        Drift rate.
    dt : float
        Time step (in ms).
    w : float
        Drift noise.
    B : float
        Decision bound.

    Returns
    -------
    C : float
        Decision variable at bound.
    z : float
        Reaction time (in ms)
    x : ndarray
        Accumulated evidence.
    '''

    # While decision variable C is within decision boundary,
    # update C according to the provided equation
    x = []
    while np.abs(C) < B:
        C = C + v * dt + w * np.random.randn()
        x.append(C)
    C = B if C > 0 else -B
    return C, len(x), np.array(x)
```

Try running 1000 simulations to get a good sense of how the model behaves. Keep the same parameters so that the only thing that differs across iterations is the noise. With positive evidence $v = 2.5$, the "correct" choice is made when the model reaches the positive bound $B = 1$; if the model reaches the negative bound, this indicates an "incorrect" choice. Plot the time series of accumulated evidence for several model runs (e.g. 10) to include both correct and incorrect trials. Plot model runs that terminate in a correct decision in one color, and model runs that terminate in an incorrect decision in another color. Why does the model occasionally make decision errors?

The model occasionally makes decision errors because of random (drift) noise that is being added at each time step. This noise leads to fluctuations in the decision varibale (C).

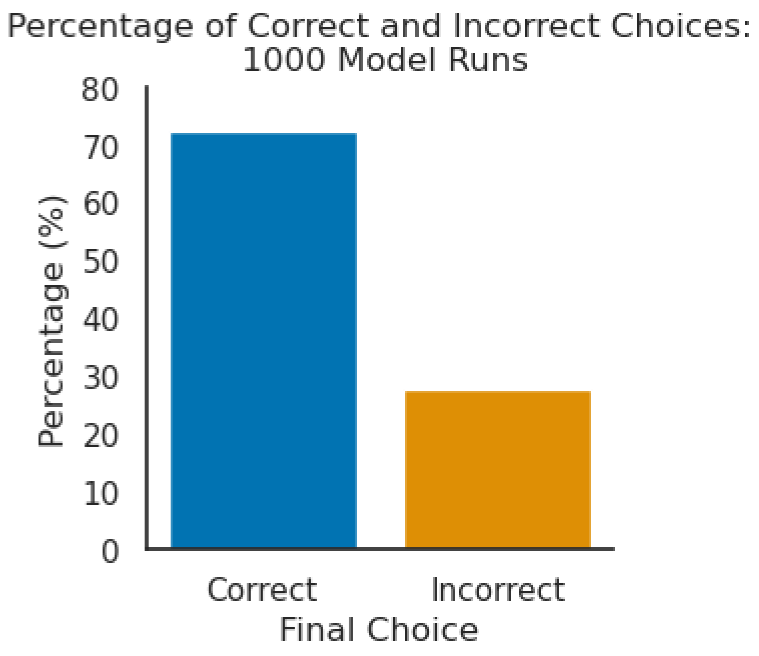
In [3]:
```
# Set a random seed
np.random.seed(1312)

# Set parameters
C0, v, dt, w, B = 0., 2.5, 5e-4, .05, 1.

# Loop through 1000 runs of the model:
results = []
for _ in range(1000):
    C, z, x = ddm(C0, v, dt, w, B)
    results.append((C, z, x))
```

In [5]:
```
# Plot first e.g. 10 model runs:
f, ax = plt.subplots(1, 1, figsize=(10, 3))


label_correct, label_incorrect = False, False
for result in results[:10]:
    C, z, x = result
    rt = np.arange(z) * dt * 1000
    if C > 0:
        if not label_correct:
            ax.plot(rt, x, color=cpal[0], label='Correct')
            label_correct = True
        else:
            ax.plot(rt, x, color='steelblue')
    else:
        if not label_incorrect:
            ax.plot(rt, x, color=cpal[1], label='Incorrect')
            label_incorrect = True
        else:
            ax.plot(rt, x, color='orange')
sns.despine()
ax.legend(title='Final Decision', loc='lower right')
ax.set_xlabel('Time (ms)')
ax.set_ylabel('Accumulated Evidence\n (Decision Value)')
ax.set_title('Accumulated Data vs. Time:\n First 10 Model Runs')
f.show()
```



Plot the proportion of runs that ended in correct and incorrect choices (e.g. a simple bar plot).

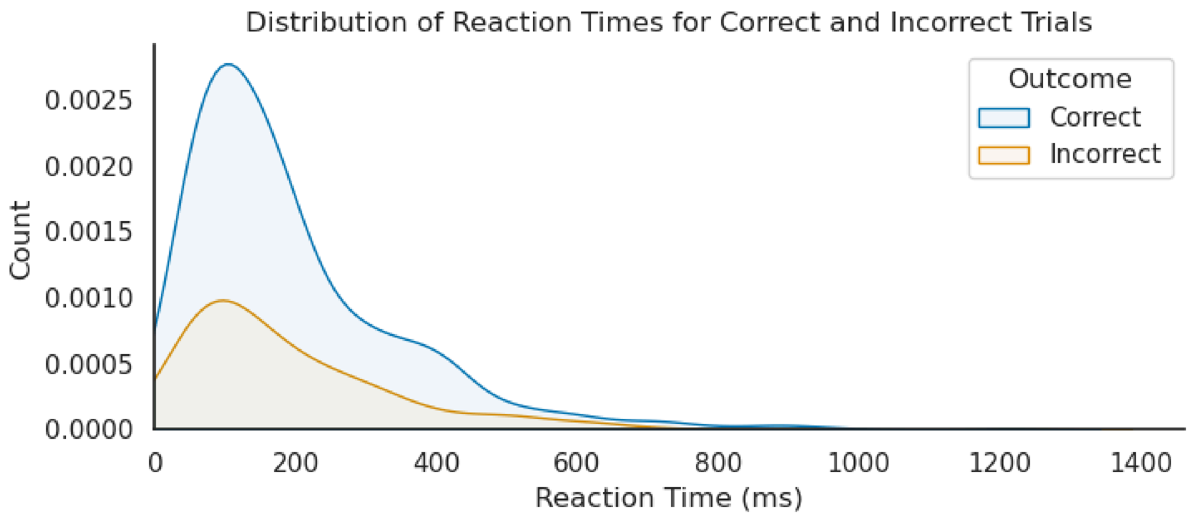
In [6]:
```
# Calculate proportions
n_correct = sum(1 for result in results if result[0] > 0)
n_incorrect = sum(1 for result in results if result[0] < 0)
proportions = [n_correct / len(results) * 100, n_incorrect / len(results

# Plot the proportion of correct and incorrect choices
f, ax = plt.subplots(1, 1, figsize=(3, 3))
ax.bar(['Correct', 'Incorrect'], proportions, color=[cpal[0], cpal[1]])
ax.set_title('Percentage of Correct and Incorrect Choices:\n 1000 Model R
ax.set_ylabel('Percentage (%)')
ax.set_xlabel('Final Choice')
ax.set_yticks(np.arange(0, 90, 10))
sns.despine()
f.show()
```



Plot the two distributions of reaction times (RTs; e.g. using `sns.histplot` or `sns.kdeplot`) for correct and incorrect trials. (You may want to convert the reaction times and outcomes to a `pandas DataFrame` before plotting with `seaborn`.) Report the mean and median of these distributions. Are they symmetric or skewed?

They are right skewed.

In [7]:
```
# Plot RT distributions of correct and incorrect trials:
df = pd.DataFrame({
    'Outcome': ['Correct' if result[0] > 0 else 'Incorrect' for result in
    'RT': [result[1] * dt * 1000 for result in results]
})

f, ax = plt.subplots(1, 1, figsize=(8, 3))
sns.kdeplot(data=df, x='RT', hue='Outcome', fill=True, alpha=0.05)
ax.set_title('Distribution of Reaction Times for Correct and Incorrect T
ax.set_xlabel('Reaction Time (ms)')
ax.set_ylabel('Count')
ax.set_xlim(0)
sns.despine()
f.show()
```



In [8]:
```
from scipy.stats import skew

# Report the mean, median, and skew:
results_mean = df.groupby('Outcome')['RT'].mean()
results_median = df.groupby('Outcome')['RT'].median()
stats_df = pd.DataFrame({
    'Mean RT (ms)': results_mean,
    'Median RT (ms)': results_median
}).reset_index()

# Determine skew by comparing mean and median
stats_df['Skew'] = np.where(stats_df['Mean RT (ms)'] > stats_df['Median 
                            'Right-skewed', 'Left-skewed')

stats_df
```

Out[8]:

	Outcome	Mean RT (ms)	Median RT (ms)	Skew
0	Correct	194.706777	148.5	Right-skewed
1	Incorrect	193.929603	145.0	Right-skewed

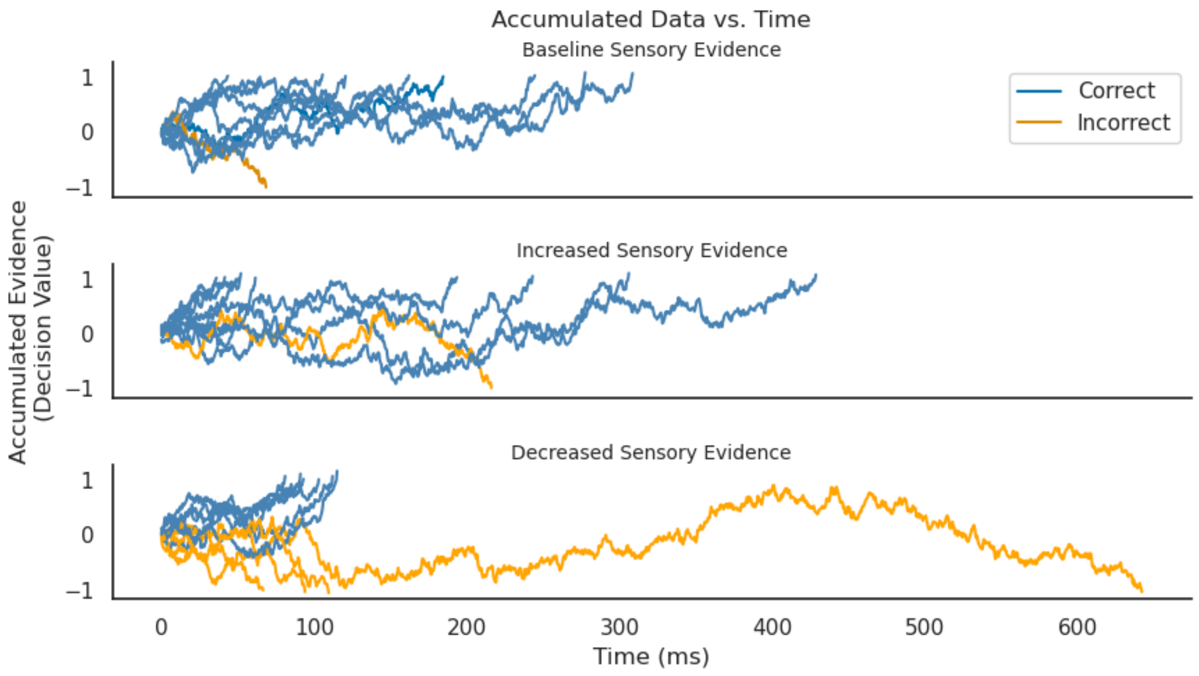
Problem 2: Varying sensory evidence

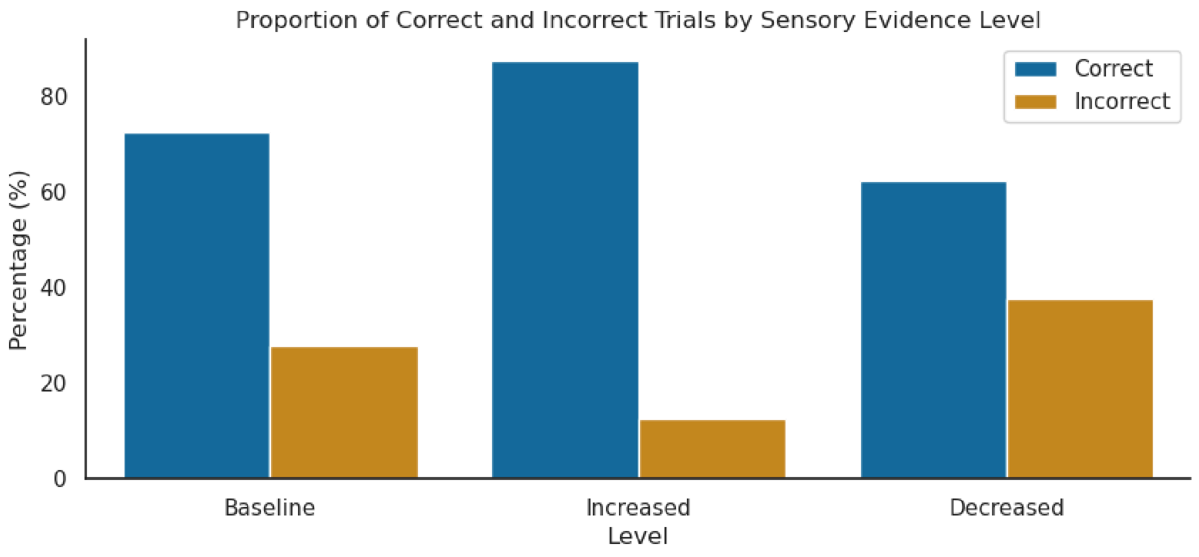
Now that we have a grasp on how the drift-diffusion model behaves, let's start varying the input. Start with the previous model with evidence v = 2.5, then run two additional models with increased $2 \times v$ evidence and decreased $.5 \times v$ evidence. As in the previous exercises, plot a few model runs, the proportion of correct and incorrect trials, and the reaction time distributions for all three levels of sensory evidence (baseline, increased, and decreased). How do these different amounts of sensory evidence affect the decision process?

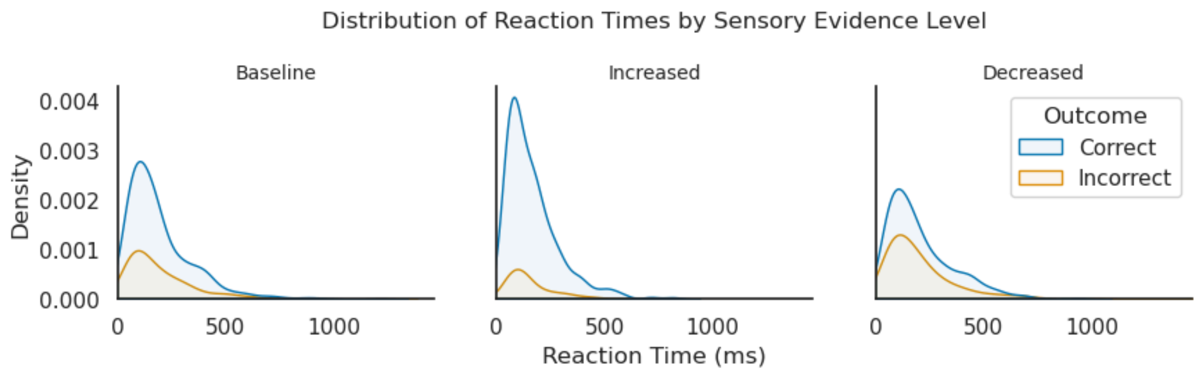
Overall, the number of incorrect decisions scales with sensory evidence. In other words, an increase in sensory evidence corresponds to an increase in accuracy.

In [9]:
```
# Set a random seed
np.random.seed(1312)

# Set up different evidence levels
C0, dt, w, B = 0., 5e-4, .05, 1.
v_base = 2.5
vs = [v_base, v_base * 2, v_base * .5]
labels = ['baseline', 'increased', 'decreased']

# Run models and plot for three evidence levels:
results, results_df = [], []
for v, label in zip(vs, labels):
    model_results = []
    for _ in range(1000):
        C, z, x = ddm(C0, v, dt, w, B)
        results_df.append({'Level': label.capitalize(), 'Outcome': 'Corre
        model_results.append((C, z, x))
    results.append(model_results)
results_df = pd.DataFrame(results_df)


f, ax = plt.subplots(3, 1, figsize=(10, 5), sharex=True, sharey=True)
f.subplots_adjust(hspace=0.5)
label_correct, label_incorrect = False, False
for i, (model_result, label) in enumerate(zip(results, labels)):
    for result in model_result[:10]:
        C, z, x = result
        timesteps = np.arange(z) * dt * 1000
        if C > 0:
            if not label_correct:
                ax[i].plot(timesteps, x, color=cpal[0], label='Correct')
                label_correct = True
            else:
                ax[i].plot(timesteps, x, color='steelblue')
        else:
            if not label_incorrect:
                ax[i].plot(timesteps, x, color=cpal[1], label='Incorrect
                label_incorrect = True
            else:
                ax[i].plot(timesteps, x, color='orange')
    ax[i].text(0.5, 1.02, label.capitalize() + " Sensory Evidence", trans
                  fontsize=10, va='bottom', ha='center')
ax[0].legend()
ax[0].set_title('Accumulated Data vs. Time\n')
ax[1].set_ylabel('Accumulated Evidence\n (Decision Value)')
ax[2].set_xlabel('Time (ms)')
sns.despine()
f.show()

# Plot proportions
df_prop = results_df.groupby('Level')['Outcome'].value_counts(normalize=
df_prop['Percentage'] *= 100
f, ax = plt.subplots(1, 1, figsize=(10, 4))
ax = sns.barplot(data=df_prop, x='Level',  y='Percentage', hue='Outcome'
ax.legend(loc='upper right')
ax.set_title('Proportion of Correct and Incorrect Trials by Sensory Evide
ax.set_ylabel('Percentage (%)')
```

```python
f.show()
sns.despine()

# Plotting the distributions of reaction times for correct and incorrect
f, ax = plt.subplots(1, 3, figsize=(10, 2), sharey=True, sharex=True)
for i, label in enumerate(labels):
    df = results_df[results_df['Level']==label.capitalize()]
    legend = True if i==2 else False
    sns.kdeplot(data=df, x='RT', hue='Outcome', fill=True, alpha=0.05, a
    ax[i].set_xlabel("")
    ax[i].text(0.5, 1.02, label.capitalize(), transform=ax[i].transAxes,
               fontsize=10, va='bottom', ha='center')
    ax[i].set_xlim(0)
ax[1].set_title('Distribution of Reaction Times by Sensory Evidence Leve
ax[1].set_xlabel('Reaction Time (ms)')
ax[0].set_ylabel('Density')
sns.despine()
f.show()
```







In [11]:

```python
# Report the mean, median, and skew:
results_mean = results_df.groupby(['Level', 'Outcome'])['RT'].mean()
results_median = results_df.groupby(['Level', 'Outcome'])['RT'].median()
stats_df = pd.DataFrame({
    'Mean RT (ms)': results_mean,
    'Median RT (ms)': results_median
}).reset_index()
stats_df['Skew'] = np.where(stats_df['Mean RT (ms)'] > stats_df['Median I
                             'Right-skewed', 'Left-skewed')
stats_df
```

Out[11]:

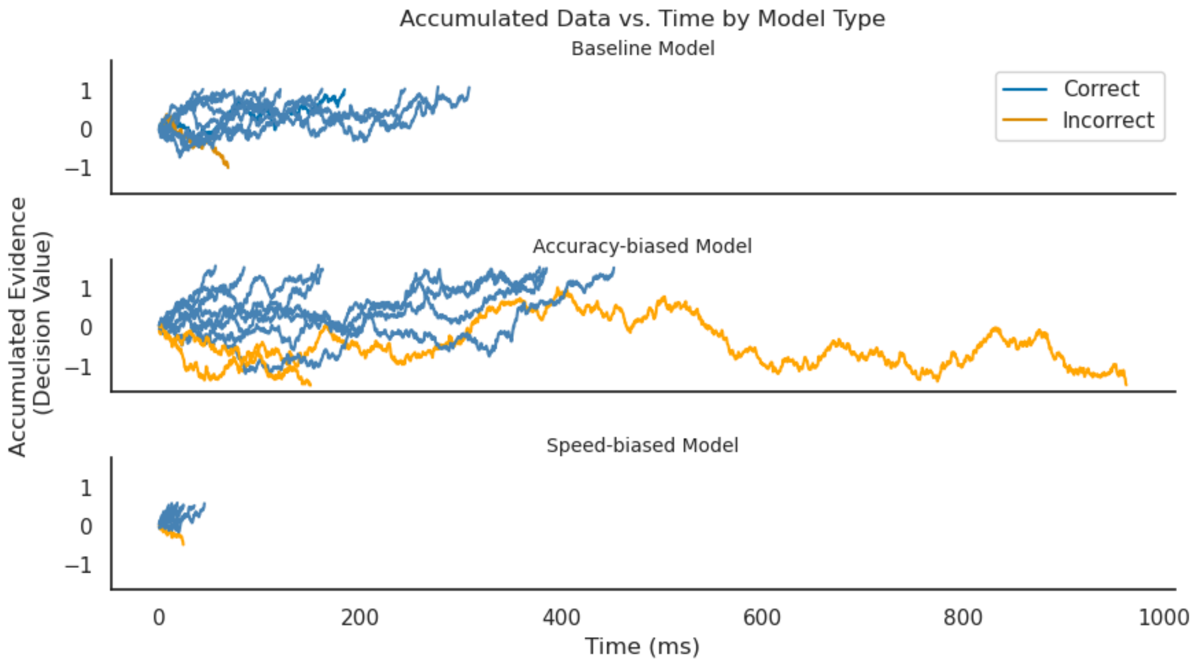
	Level	Outcome	Mean RT (ms)	Median RT (ms)	Skew
0	Baseline	Correct	194.706777	148.5	Right-skewed
1	Baseline	Incorrect	193.929603	145.0	Right-skewed
2	Decreased	Correct	206.643660	158.5	Right-skewed
3	Decreased	Incorrect	206.655172	153.5	Right-skewed
4	Increased	Correct	164.875716	128.0	Right-skewed
5	Increased	Incorrect	157.188976	119.5	Right-skewed

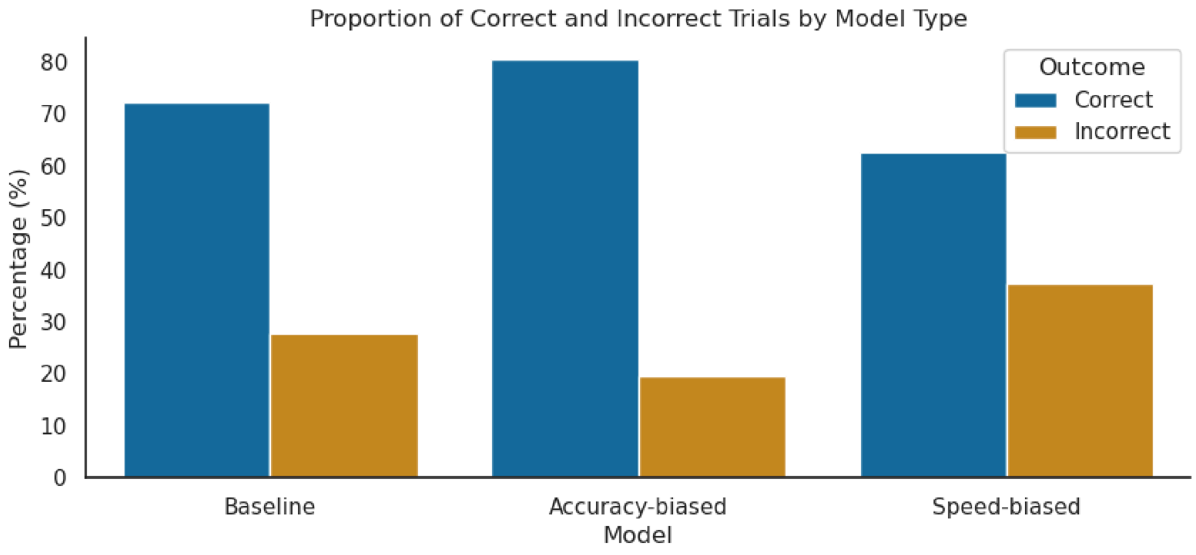
Problem 3: Speed-accuracy tradeoff

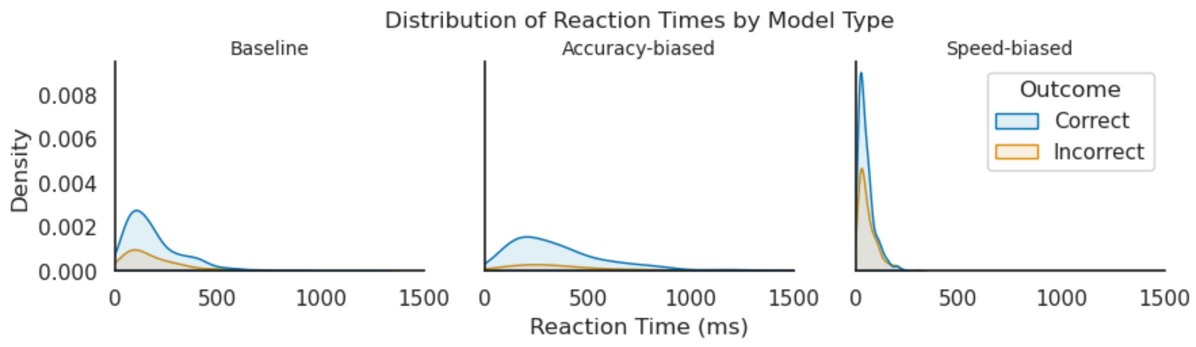
Subjects appear to be able to trade accuracy for speed in most perceptual decision-making tasks. The drift-diffusion model can capture this tradeoff in a relatively simple way. Describe how you might model this speed-accuracy tradeoff. Using a fixed sensory evidence $v = 2.5$, run three different versions of the model: the baseline model from previous exercises, an "accuracy"-biased model, and a "speed"-biased model. As in the previous exercise, plot a few model runs, the proportion of correct and incorrect trials, and the reaction time distributions for all three models (baseline, accuracy, and speed).

In [12]:
```
# Set a random seed
np.random.seed(1312)

# Set up different models:
C0, dt, w, v = 0., 5e-4, .05, 2.5
Bs = [1, 1.5, 0.5]
labels = ['baseline', 'accuracy-biased', 'speed-biased']


results, results_df = [], []
for B, label in zip(Bs, labels):
    model_results = []
    for _ in range(1000):
        C, z, x = ddm(C0, v, dt, w, B)
        results_df.append({'Model': label.capitalize(), 'Outcome': 'Corr
        model_results.append((C, z, x))
    results.append(model_results)
results_df = pd.DataFrame(results_df)


f, ax = plt.subplots(3, 1, figsize=(10, 5), sharex=True, sharey=True)
f.subplots_adjust(hspace=0.5)
label_correct, label_incorrect = False, False
for i, (model_result, label) in enumerate(zip(results, labels)):
    for result in model_result[:10]:
        C, z, x = result
        timesteps = np.arange(z) * dt * 1000
        if C > 0:
            if not label_correct:
                ax[i].plot(timesteps, x, color=cpal[0], label='Correct')
                label_correct = True
            else:
                ax[i].plot(timesteps, x, color='steelblue')
        else:
            if not label_incorrect:
                ax[i].plot(timesteps, x, color=cpal[1], label='Incorrect
                label_incorrect = True
            else:
                ax[i].plot(timesteps, x, color='orange')
    ax[i].text(0.5, 1.02, label.capitalize() + " Model", transform=ax[i]
               fontsize=10, va='bottom', ha='center')
ax[0].legend()
ax[0].set_title('Accumulated Data vs. Time by Model Type\n')
ax[1].set_ylabel('Accumulated Evidence\n (Decision Value)')
ax[2].set_xlabel('Time (ms)')
sns.despine()
f.show()

# Plot proportions
df_prop = results_df.groupby('Model')['Outcome'].value_counts(normalize=
df_prop['Percentage'] *= 100
f, ax = plt.subplots(1, 1, figsize=(10, 4))
ax = sns.barplot(data=df_prop, x='Model',  y='Percentage', hue='Outcome'
ax.set_title('Proportion of Correct and Incorrect Trials by Model Type')
ax.set_ylabel('Percentage (%)')
f.show()
sns.despine()
```

```
# Plotting the distributions of reaction times for correct and incorrect
f, ax = plt.subplots(1, 3, figsize=(10, 2), sharey=True, sharex=True)
for i, label in enumerate(labels):
    df = results_df[results_df['Model']==label.capitalize()]
    legend = True if i==2 else False
    sns.kdeplot(data=df, x='RT', hue='Outcome', fill=True, alpha=0.1, ax
    ax[i].set_xlabel("")
    ax[i].text(0.5, 1.02, label.capitalize(), transform=ax[i].transAxes,
                fontsize=10, va='bottom', ha='center')
    ax[i].set_xlim(left=0, right=1500)
ax[1].set_title('Distribution of Reaction Times by Model Type\n')
ax[1].set_xlabel('Reaction Time (ms)')
ax[0].set_ylabel('Density')
sns.despine()
f.show()
```







In [13]:

```
# Report the mean, median, and skew:
results_mean = results_df.groupby(['Model', 'Outcome'])['RT'].mean()
results_median = results_df.groupby(['Model', 'Outcome'])['RT'].median()
stats_df = pd.DataFrame({
    'Mean RT (ms)': results_mean,
    'Median RT (ms)': results_median
}).reset_index()
stats_df['Skew'] = np.where(stats_df['Mean RT (ms)'] > stats_df['Median I
                             'Right-skewed', 'Left-skewed')
stats_df
```

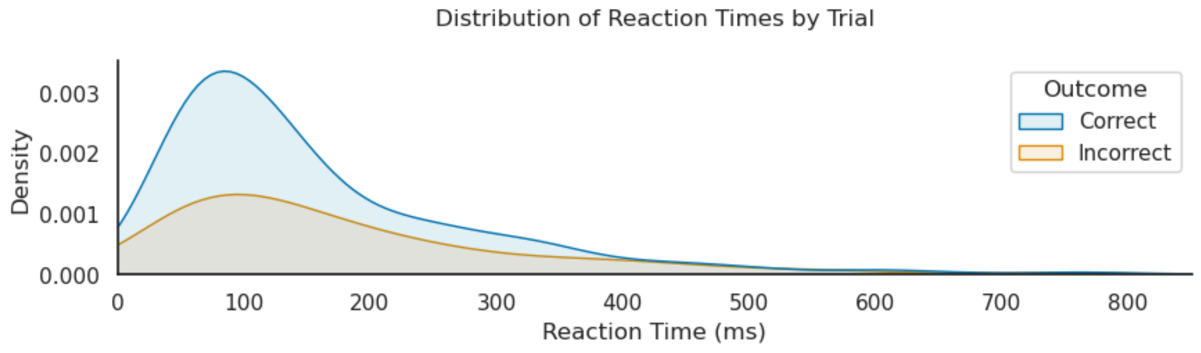
Out[13]:

	Model	Outcome	Mean RT (ms)	Median RT (ms)	Skew
0	Accuracy-biased	Correct	385.713043	307.5	Right-skewed
1	Accuracy-biased	Incorrect	444.102564	330.5	Right-skewed
2	Baseline	Correct	194.706777	148.5	Right-skewed
3	Baseline	Incorrect	193.929603	145.0	Right-skewed
4	Speed-biased	Correct	50.810703	39.0	Right-skewed
5	Speed-biased	Incorrect	57.512032	41.5	Right-skewed

Problem 4: Trial-by-trial sensory variability

In the original "baseline" model, we see that the reaction time distributions for correct and incorrect trials are largely overlapping. However, in experiments, reaction times for incorrect trials are typically longer than for correct trials. To capture this effect, let's add some trial-by-trial variability: instead of a fixed v for all trials, allow v to vary across trials according to a normal distribution $v = 2.5 + \mathcal{N}(0, 5)$. Report the mean, median, and skew as in Problem 1.

In [16]:
```
# Set a random seed
np.random.seed(1312)

# Run models with trial-by-trial variability:
C0, dt, w, B = 0., 5e-4, .05, 1.

# Plot RT distributions of correct and incorrect trials:
results_df = []
for _ in range(1000):
    v = 2.5 + np.random.normal(0, 5)
    C, z, x = ddm(C0, v, dt, w, B)
    results_df.append({'Trial': label.capitalize(), 'Outcome': 'Correct'
results_df = pd.DataFrame(results_df)

f, ax = plt.subplots(1, 1, figsize=(10, 2), sharey=True, sharex=True)
sns.kdeplot(data=results_df, x='RT', hue='Outcome', fill=True, alpha=0.1
ax.set_xlabel("Reaction Time (ms)")
ax.set_title('Distribution of Reaction Times by Trial\n')
ax.set_xlabel('Reaction Time (ms)')
ax.set_ylabel('Density')
ax.set_xlim(left=0, right=850)
sns.despine()
f.show()
```



In [17]:
```
# Report the mean, median, and skew:
results_mean = results_df.groupby('Outcome')['RT'].mean()
results_median = results_df.groupby('Outcome')['RT'].median()
stats_df = pd.DataFrame({
    'Mean RT (ms)': results_mean,
    'Median RT (ms)': results_median
}).reset_index()
stats_df['Skew'] = np.where(stats_df['Mean RT (ms)'] > stats_df['Median R
                            'Right-skewed', 'Left-skewed')
stats_df
```

Out[17]:

	Outcome	Mean RT (ms)	Median RT (ms)	Skew
0	Correct	158.269817	115.75	Right-skewed
1	Incorrect	184.327035	134.75	Right-skewed