Linear Regression

with Categories

Incorporating Categories

Categories

- Goal: Model causes that are not continuous

Categories

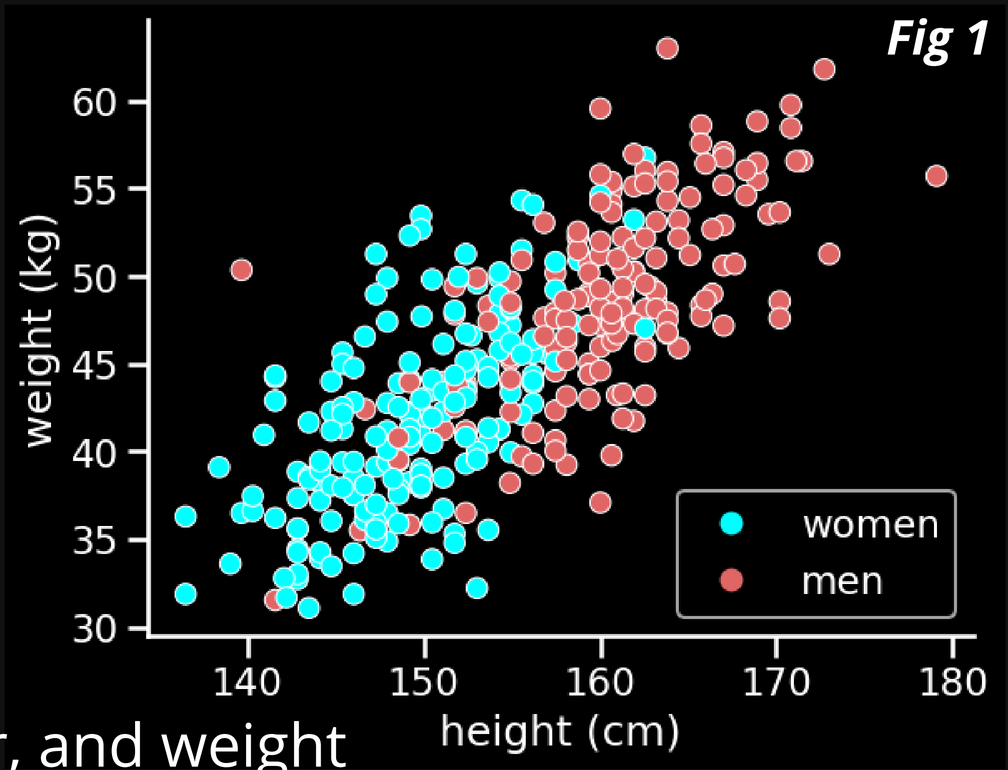
- Goal: Model causes that are not continuous
- Categories: discrete, unordered types

Categories

- Goal: Model causes that are not continuous
- Categories: discrete, unordered types
- Approach: **stratify** by categories

Categories

- Goal: Model causes that are not continuous
- Categories: discrete, unordered types
- Approach: **stratify** by categories
  - fit separate line for each category



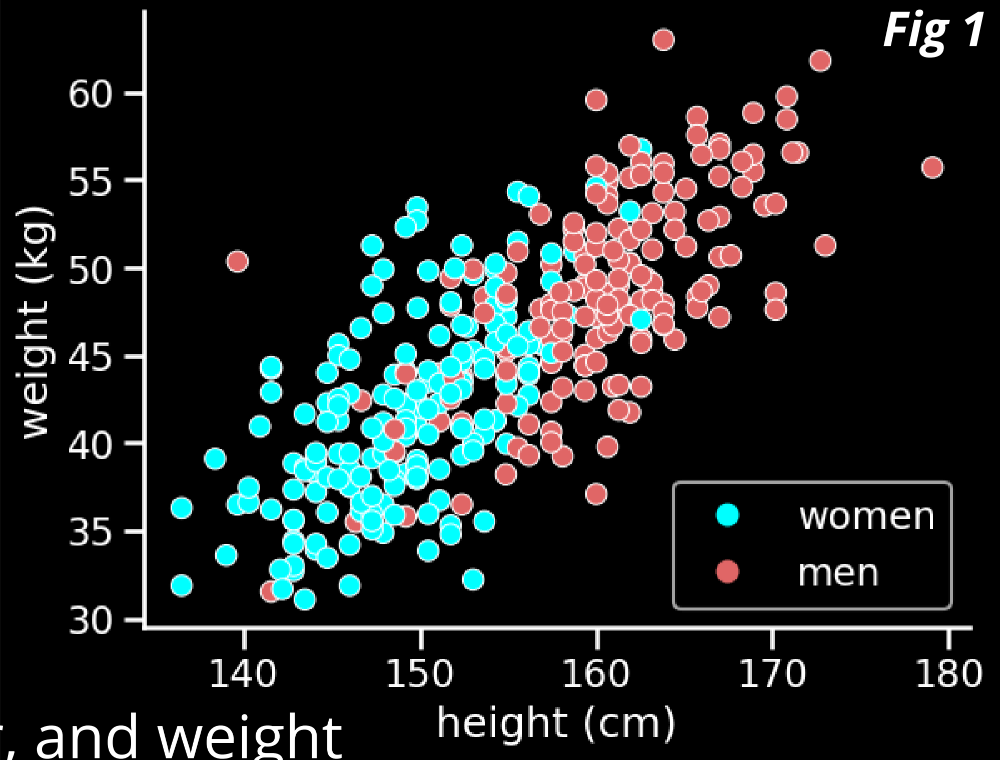
Adult height, gender, and weight

Code 1

```
import pandas as pd

df = pd.read_csv(
    "Data/Howell1.csv",
    sep=';',
    header=0
)
df2 = df[df.age >= 18]
df2.head()
```

	height	weight	age	male
0	151.765	47.825606	63.0	1
1	139.700	36.485807	63.0	0
2	136.525	31.864838	65.0	0
3	156.845	53.041914	41.0	1
4	145.415	41.276872	51.0	0

Fig 1



Adult height, gender, and weight

Code 1

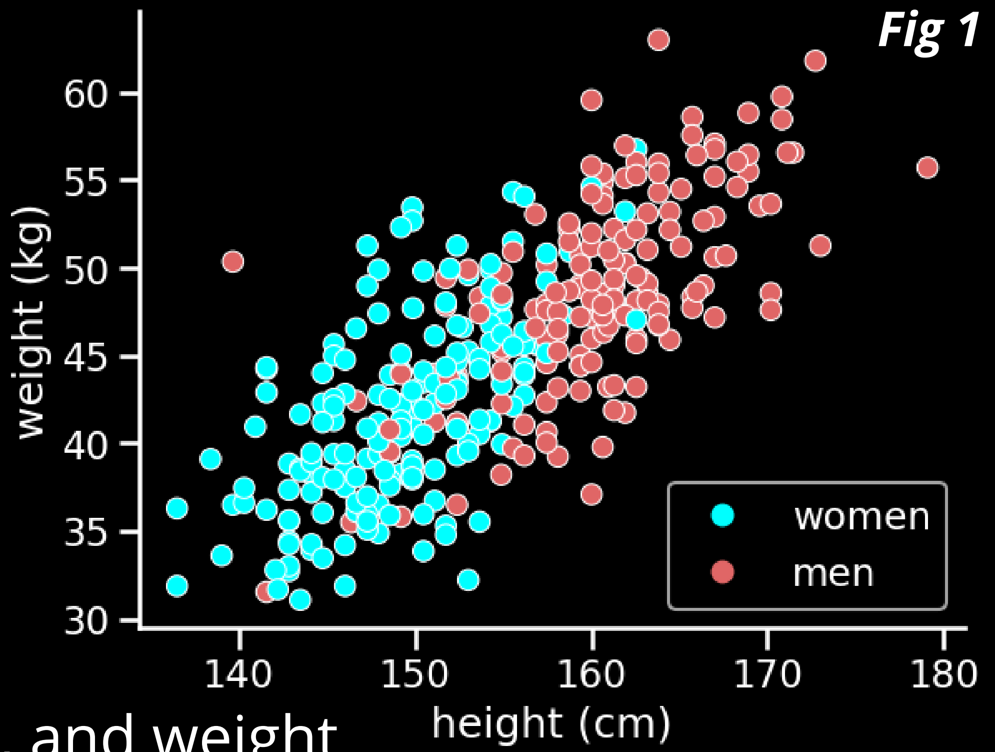
```
import pandas as pd

df = pd.read_csv(
    "Data/Howell1.csv",
    sep=';',
    header=0
)
df2 = df[df.age >= 18]
df2.head()
```

	height	weight	age	male
0	151.765	47.825606	63.0	1
1	139.700	36.485807	63.0	0
2	136.525	31.864838	65.0	0
3	156.845	53.041914	41.0	1
4	145.415	41.276872	51.0	0

Code 2

```
import seaborn as sns
from matplotlib.lines import Line2D

ax = sns.scatterplot(data=df2, x="height", y="weight", hue="male",
        palette=["cyan", "#e06666"]
)

custom = [Line2D([], [], marker='o', color="cyan", linestyle='None'),
          Line2D([], [], marker='o', color="#e06666", linestyle='None')]

_ = plt.legend(custom, ['women', 'men'], loc='lower right')

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("weight (kg)")
```

Fig 1



Adult height, gender, and weight

generates points

- uses ***height*** column for x values
- uses ***weight*** column for y values
- uses ***male*** column to color ("hue") points

Code 1

```python
import pandas as pd

df = pd.read_csv(
    "Data/Howell1.csv",
    sep=';',
    header=0
)
df2 = df[df.age >= 18]
df2.head()
```

	height	weight	age	male
0	151.765	47.825606	63.0	1
1	139.700	36.485807	63.0	0
2	136.525	31.864838	65.0	0
3	156.845	53.041914	41.0	1
4	145.415	41.276872	51.0	0

Code 2

```python
import seaborn as sns
from matplotlib.lines import Line2D

ax = sns.scatterplot(data=df2, x="height", y="weight", hue="male",
        palette=["cyan", "#e06666"]
)

custom = [Line2D([], [], marker='o', color="cyan", linestyle='None'),
          Line2D([], [], marker='o', color="#e06666", linestyle='None')]

_ = plt.legend(custom, ['women', 'men'], loc='lower right')

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("weight (kg)")
```

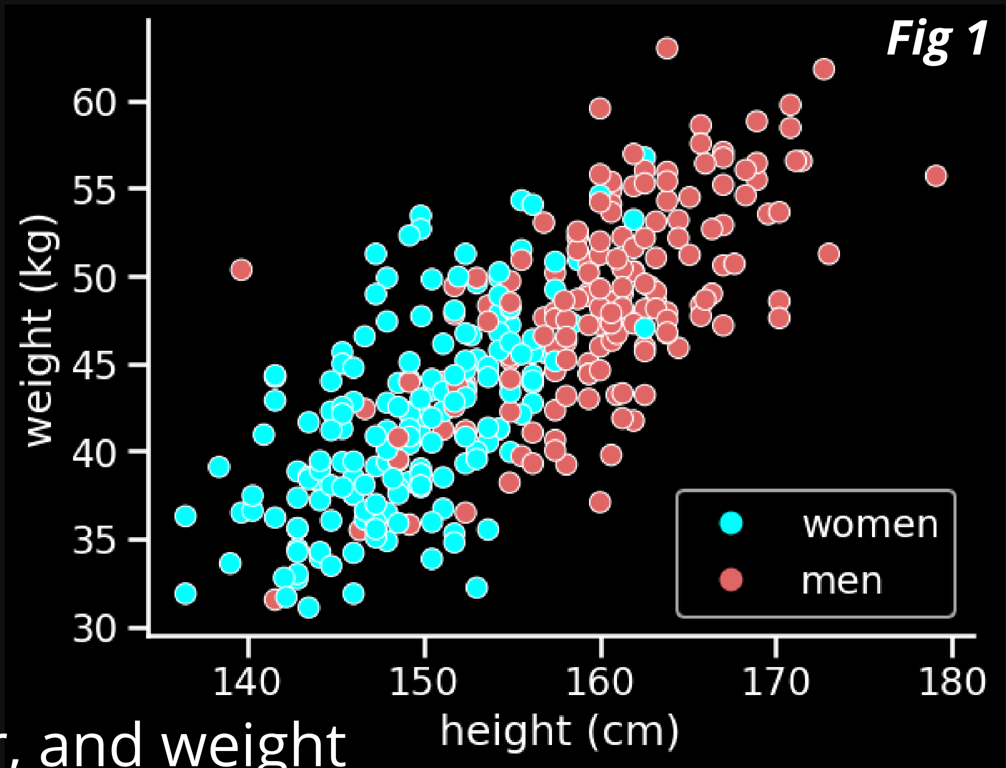


Fig 1

Adult height, gender, and weight

```python
import pandas as pd

df = pd.read_csv(
    "Data/Howell1.csv",
    sep=';',
    header=0
)
df2 = df[df.age >= 18]
df2.head()
```

Code 1

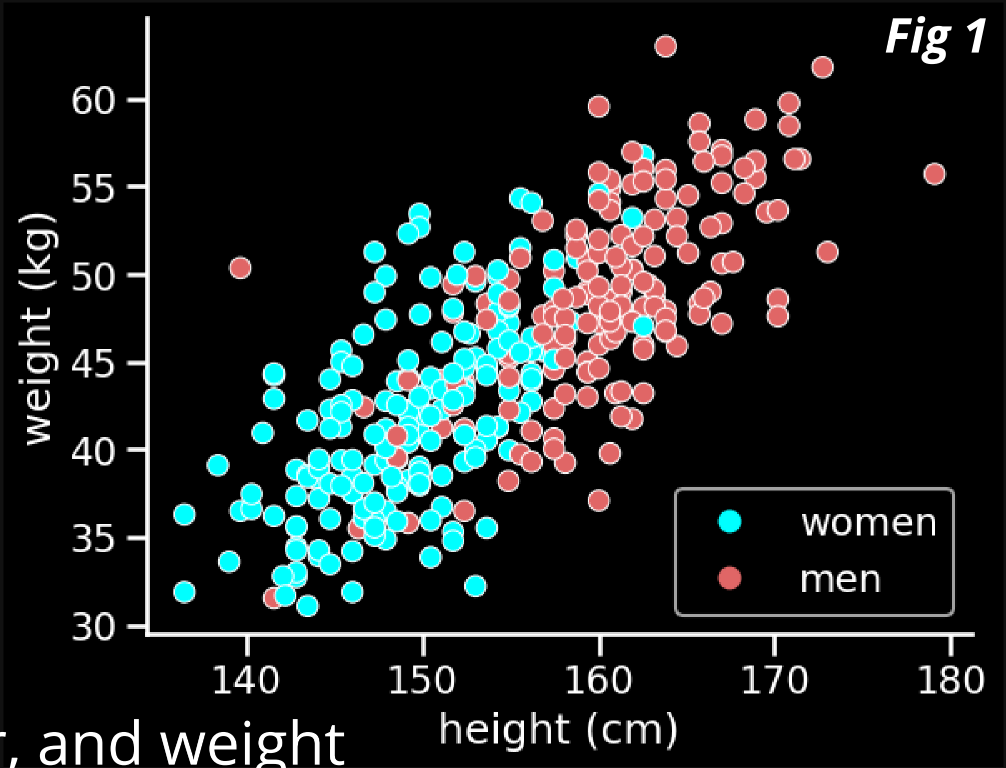
custom provides specifications for legend

	height	weight	age	male
0	151.765	47.825606	63.0	1
1	139.700	36.485807	63.0	0
2	136.525	31.864838	65.0	0
3	156.845	53.041914	41.0	1
4	145.415	41.276872	51.0	0

Code 2

```python
import seaborn as sns
from matplotlib.lines import Line2D

ax = sns.scatterplot(data=df2, x="height", y="weight", hue="male",
        palette=["cyan", "#e06666"]
)

custom = [Line2D([], [], marker='o', color="cyan", linestyle='None'),
          Line2D([], [], marker='o', color="#e06666", linestyle='None')]

_ = plt.legend(custom, ['women', 'men'], loc='lower right')

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("weight (kg)")
```

Fig 1



Adult height, gender, and weight

Think scientifically first

How are height, gender, and weight ***causally*** related?

Think scientifically first

How are height, gender, and weight ***causally*** related?

How are height, gender, and weight ***statistically*** related?

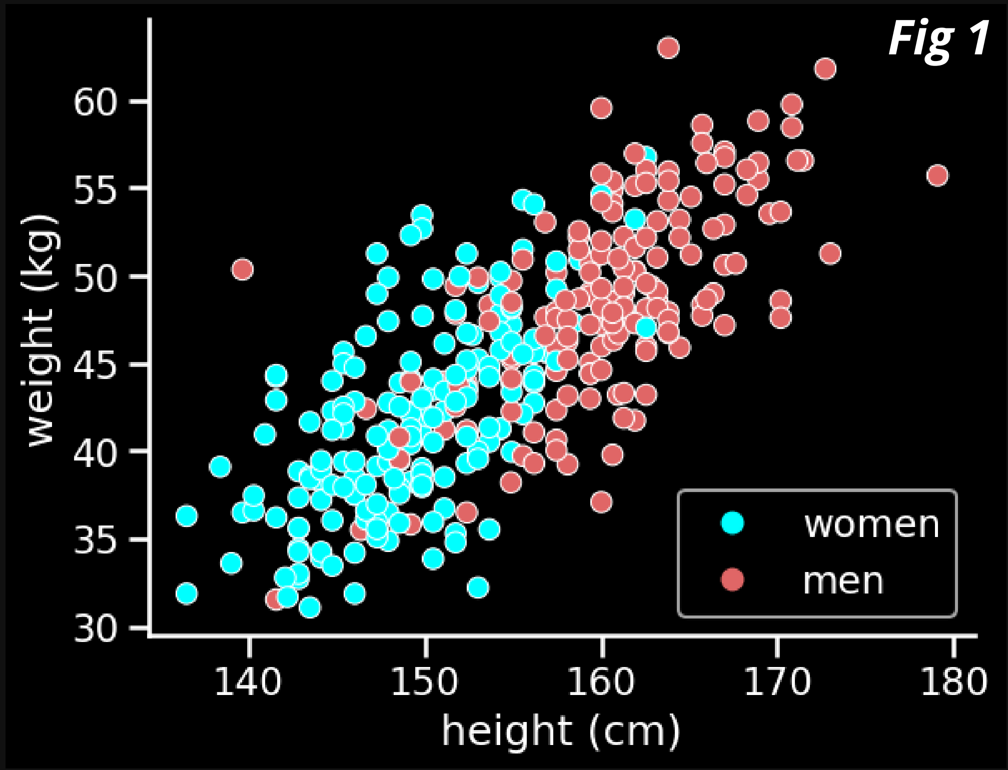
Think scientifically first

How are height, gender, and weight ***causally*** related?

How are height, gender, and weight ***statistically*** related?

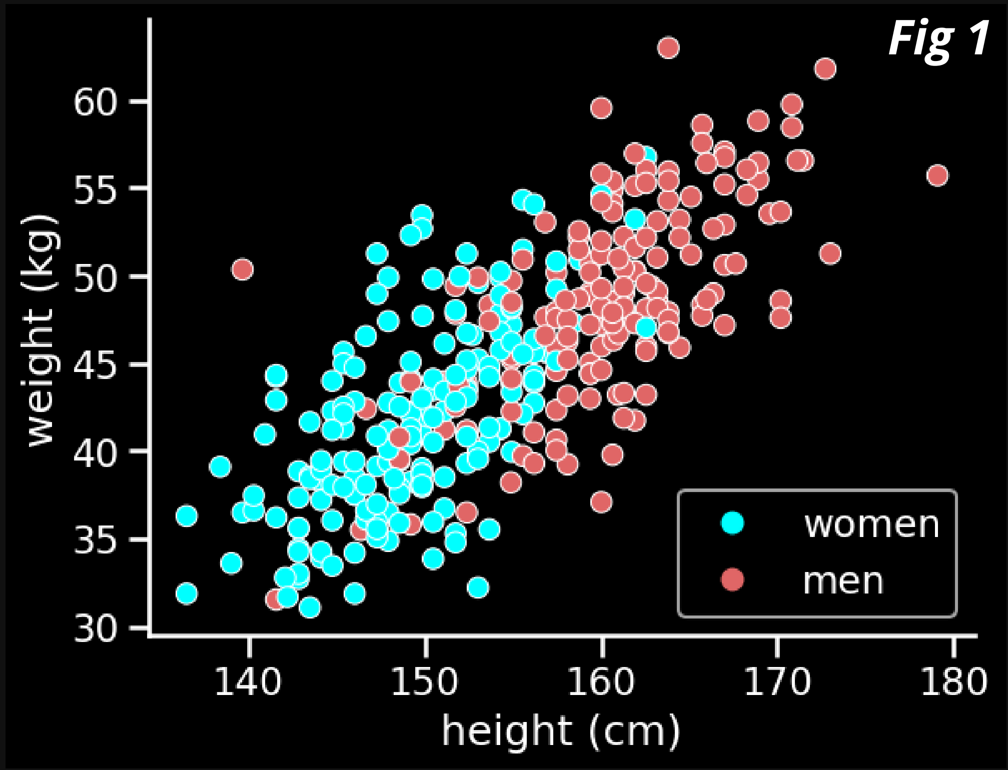
$$\begin{tikzcd} H \arrow[r] & W \\ G \arrow[u] \arrow[ur] & \end{tikzcd}$$

Causes are not included in data



Causes are not included in data

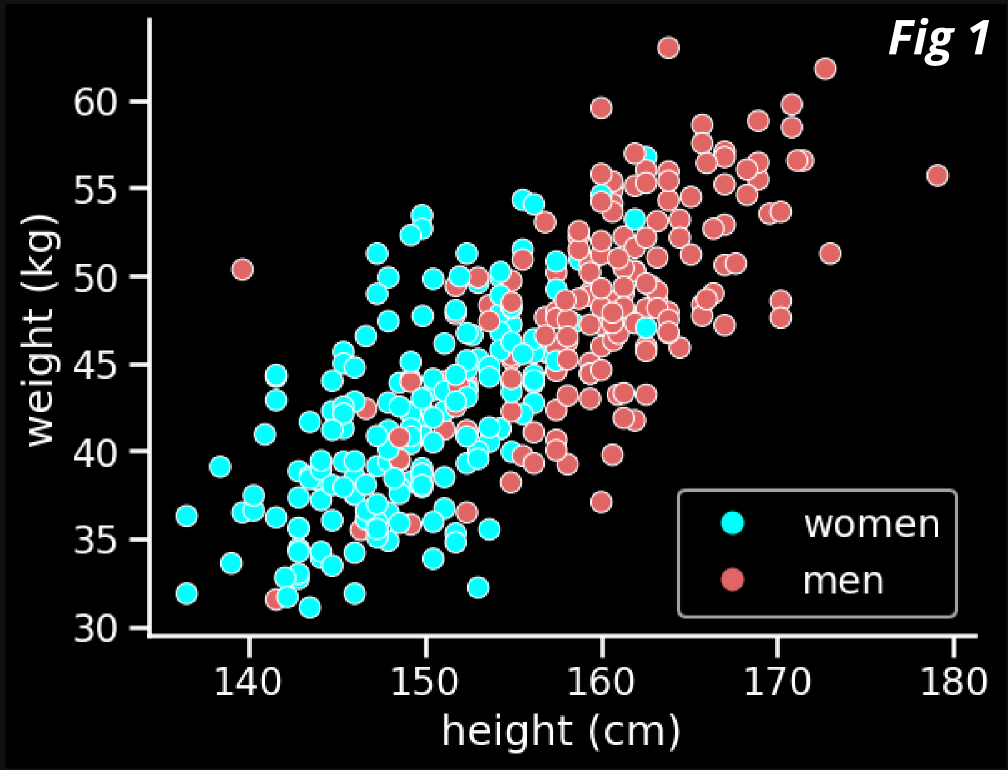
$H \longrightarrow W$



Causes are not included in data

$H \longrightarrow W$

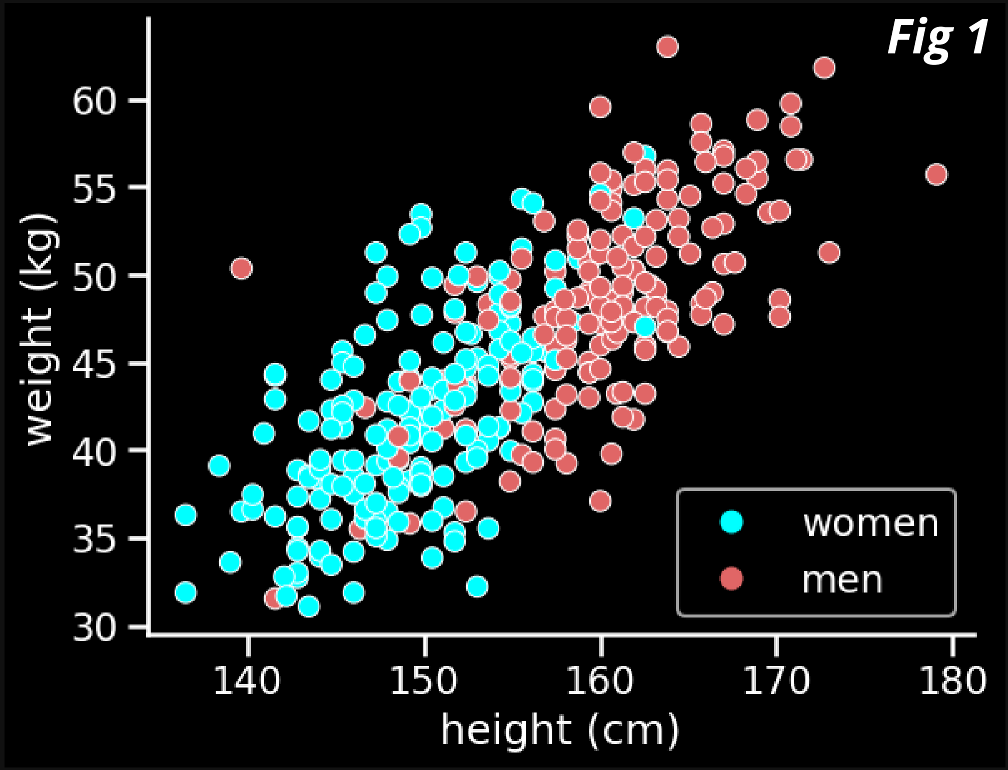
$H \longleftarrow W$



Causes are not included in data

$H \longrightarrow W$

$H \longleftarrow W$



Causes are not included in data

```
ax = sns.kdeplot(
        data=df2, x="height", hue="male",
        palette=["cyan", "#e06666"]
)
custom = [Line2D([], [], marker='_', color="cyan"),
          Line2D([], [], marker='_', color="#e06666")]

_ = plt.legend(custom, ['women', 'men'])

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("density")
_ = sns.despine()
```

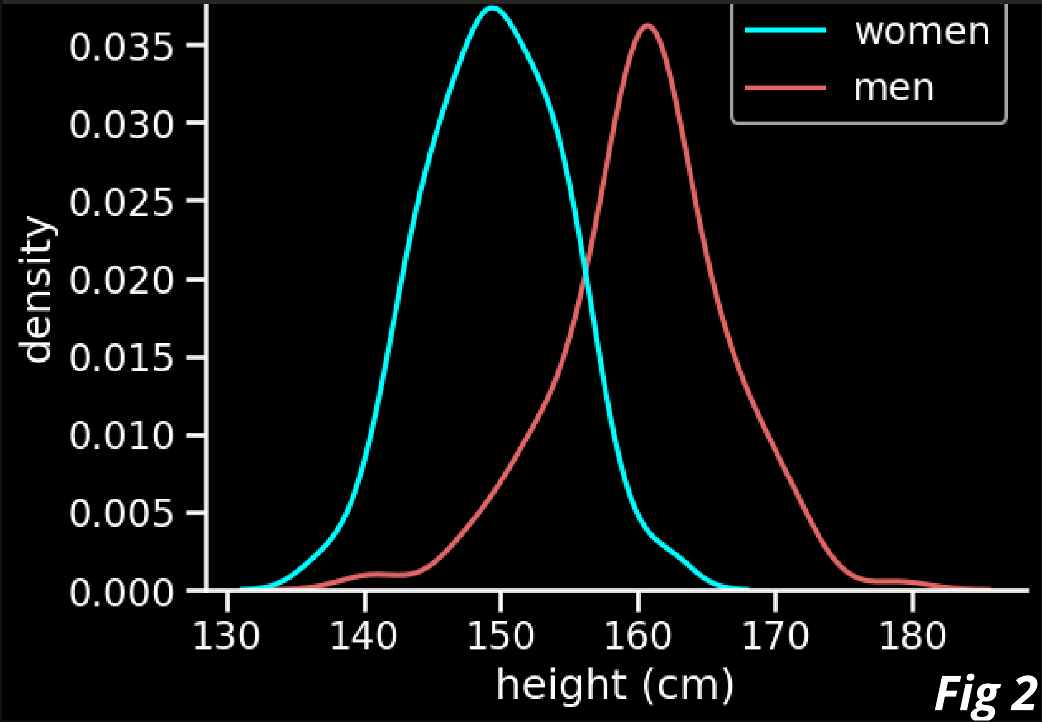


Fig 2

Causes are not included in data

$H \longrightarrow G$

```
ax = sns.kdeplot(
        data=df2, x="height", hue="male",
        palette=["cyan", "#e06666"]
)
custom = [Line2D([], [], marker='_', color="cyan"),
          Line2D([], [], marker='_', color="#e06666")]

_ = plt.legend(custom, ['women', 'men'])

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("density")
_ = sns.despine()
```

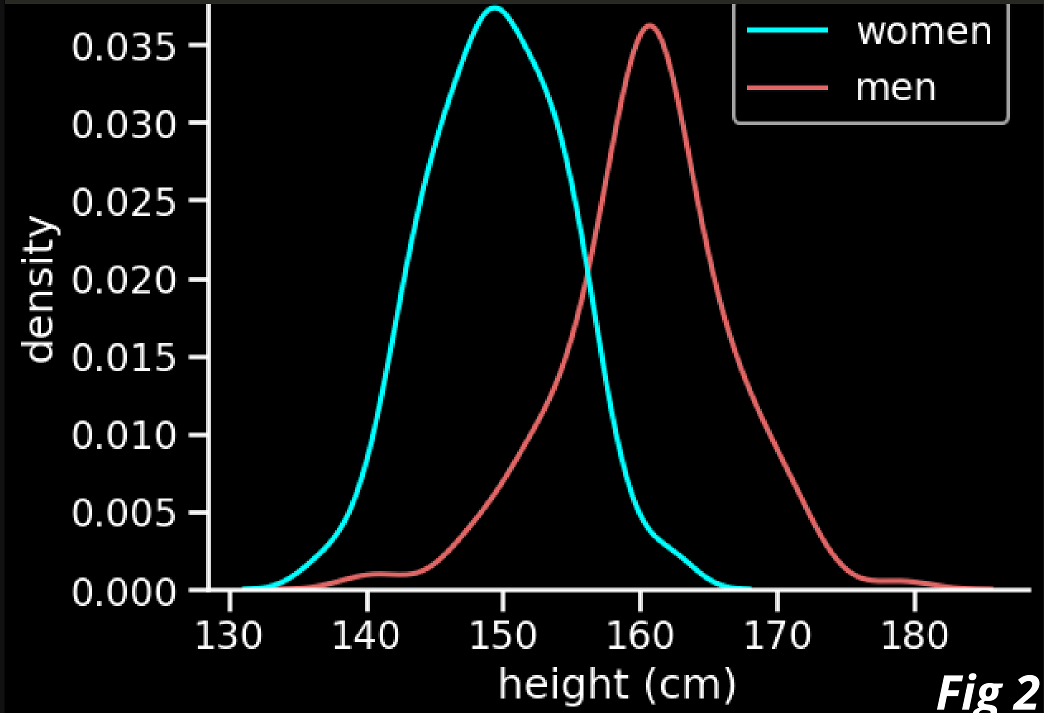


Fig 2

Causes are not included in data

$H \longrightarrow G$

$H \longleftarrow G$

```python
ax = sns.kdeplot(
        data=df2, x="height", hue="male",
        palette=["cyan", "#e06666"]
)
custom = [Line2D([], [], marker='_', color="cyan"),
          Line2D([], [], marker='_', color="#e06666")]

_ = plt.legend(custom, ['women', 'men'])

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("density")
_ = sns.despine()
```

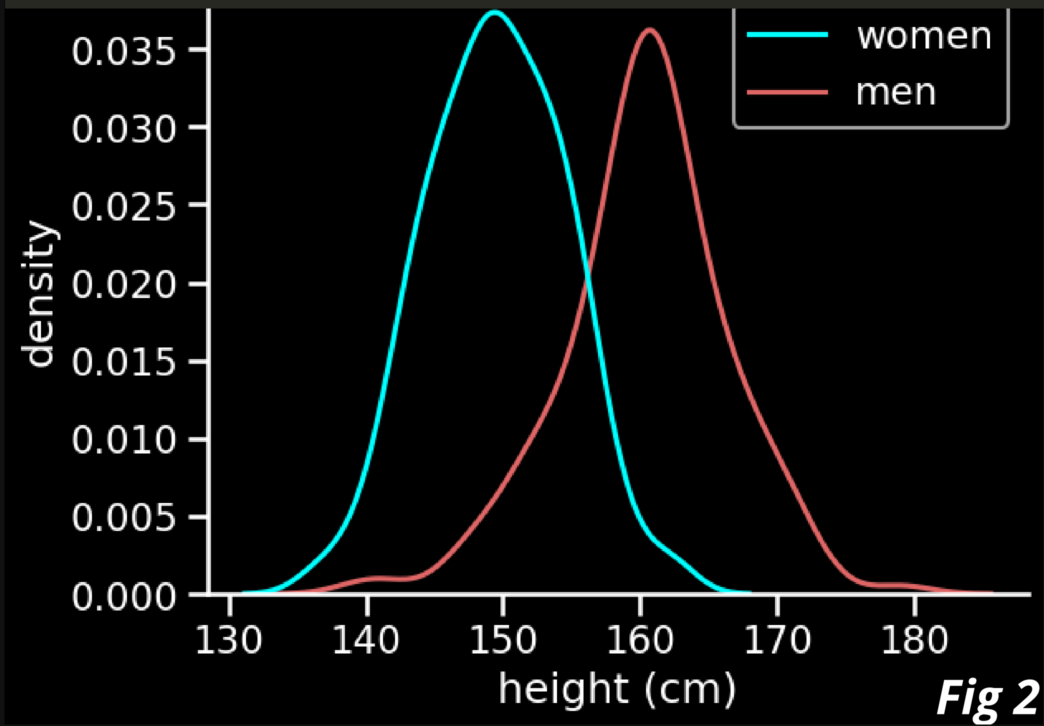


Fig 2

Causes are not included in data

$\cancel{H \longrightarrow G}$

$H \longleftarrow G$

```python
ax = sns.kdeplot(
        data=df2, x="height", hue="male",
        palette=["cyan", "#e06666"]
)
custom = [Line2D([], [], marker='_', color="cyan"),
          Line2D([], [], marker='_', color="#e06666")]

_ = plt.legend(custom, ['women', 'men'])

_ = ax.set_xlabel("height (cm)")
_ = ax.set_ylabel("density")
_ = sns.despine()
```

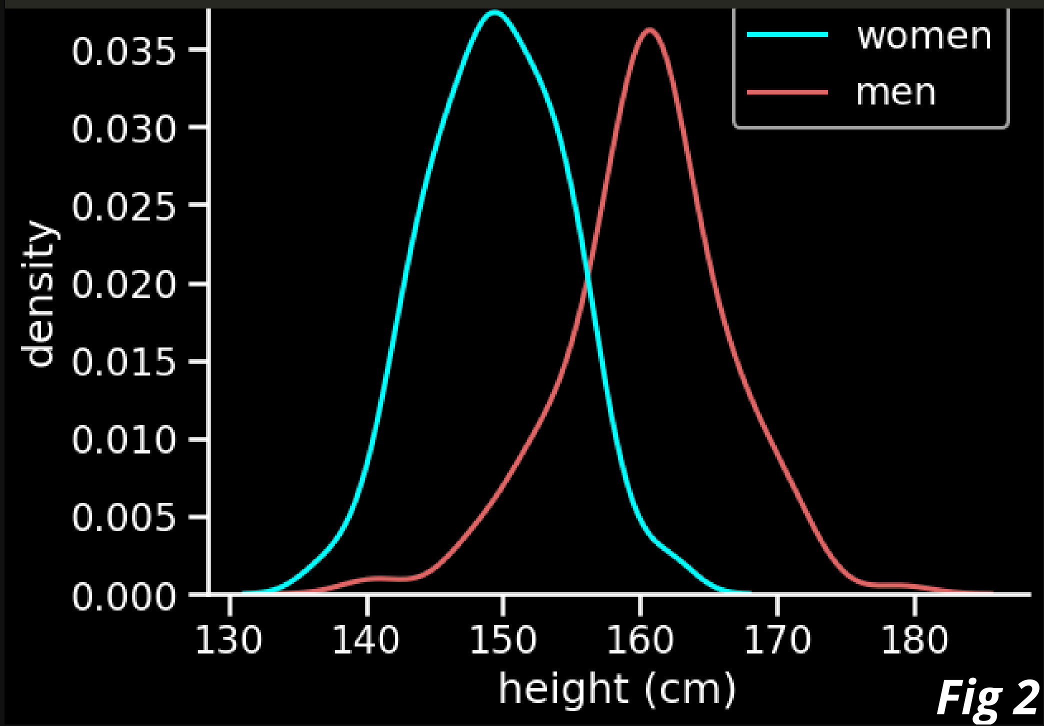


Fig 2

Causes are not included in data

$\cancel{H \rightarrow G}$

$H \leftarrow G$

$\cancel{W \rightarrow G}$

$W \leftarrow G$

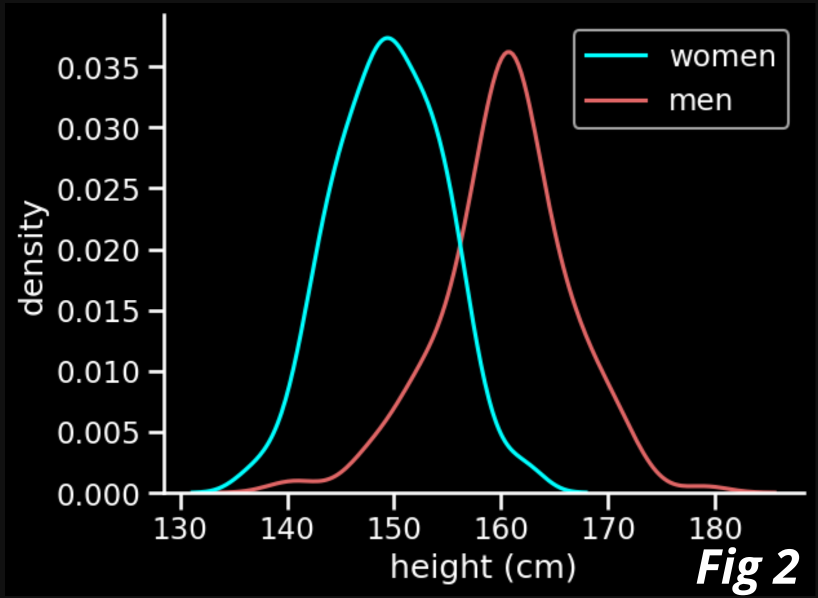


Fig 2

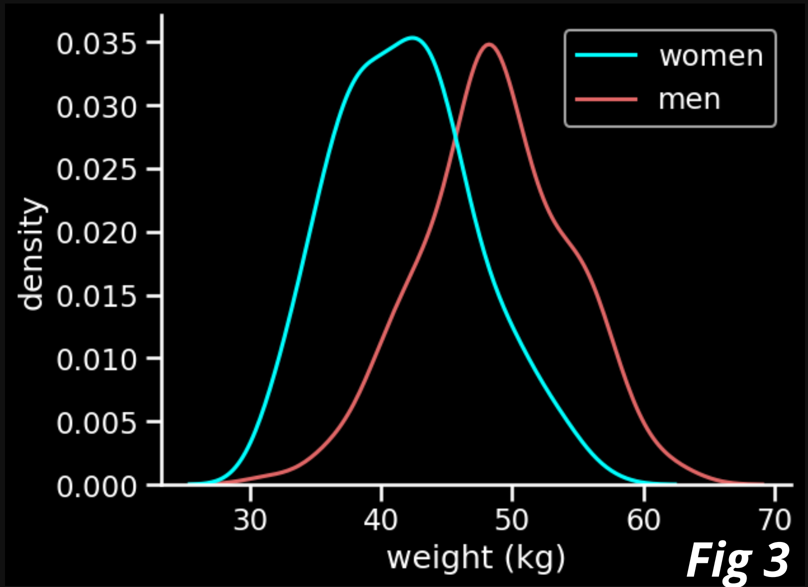
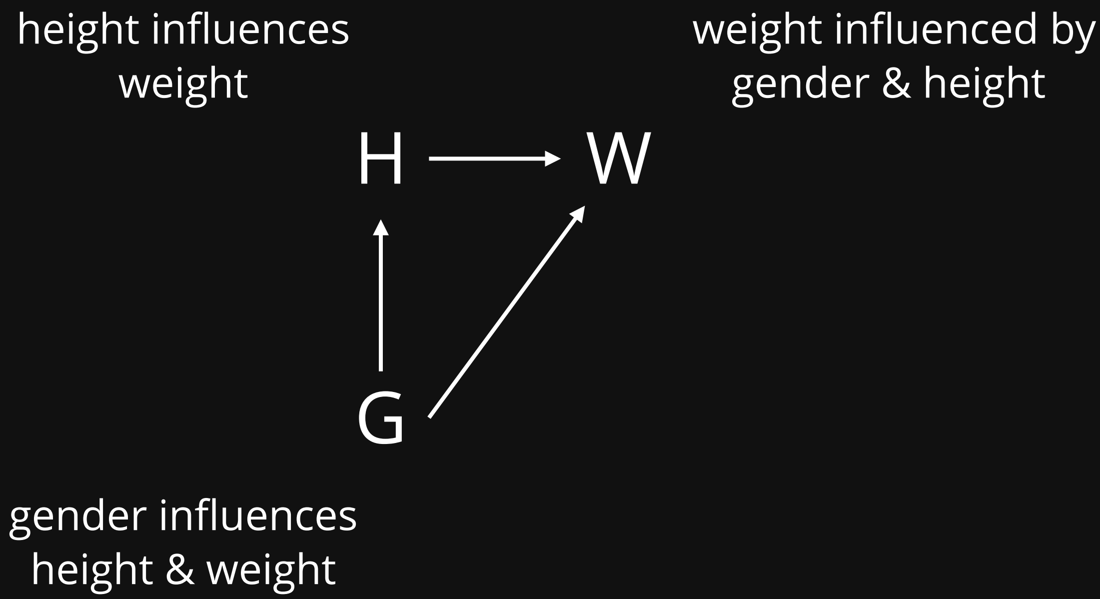


Fig 3


height influences weight
weight influenced by gender & height
H
W
G
gender influences height & weight

$$
\begin{tikzcd}
H \arrow[r] & W \\
G \arrow[u] \arrow[ur] &
\end{tikzcd}
$$

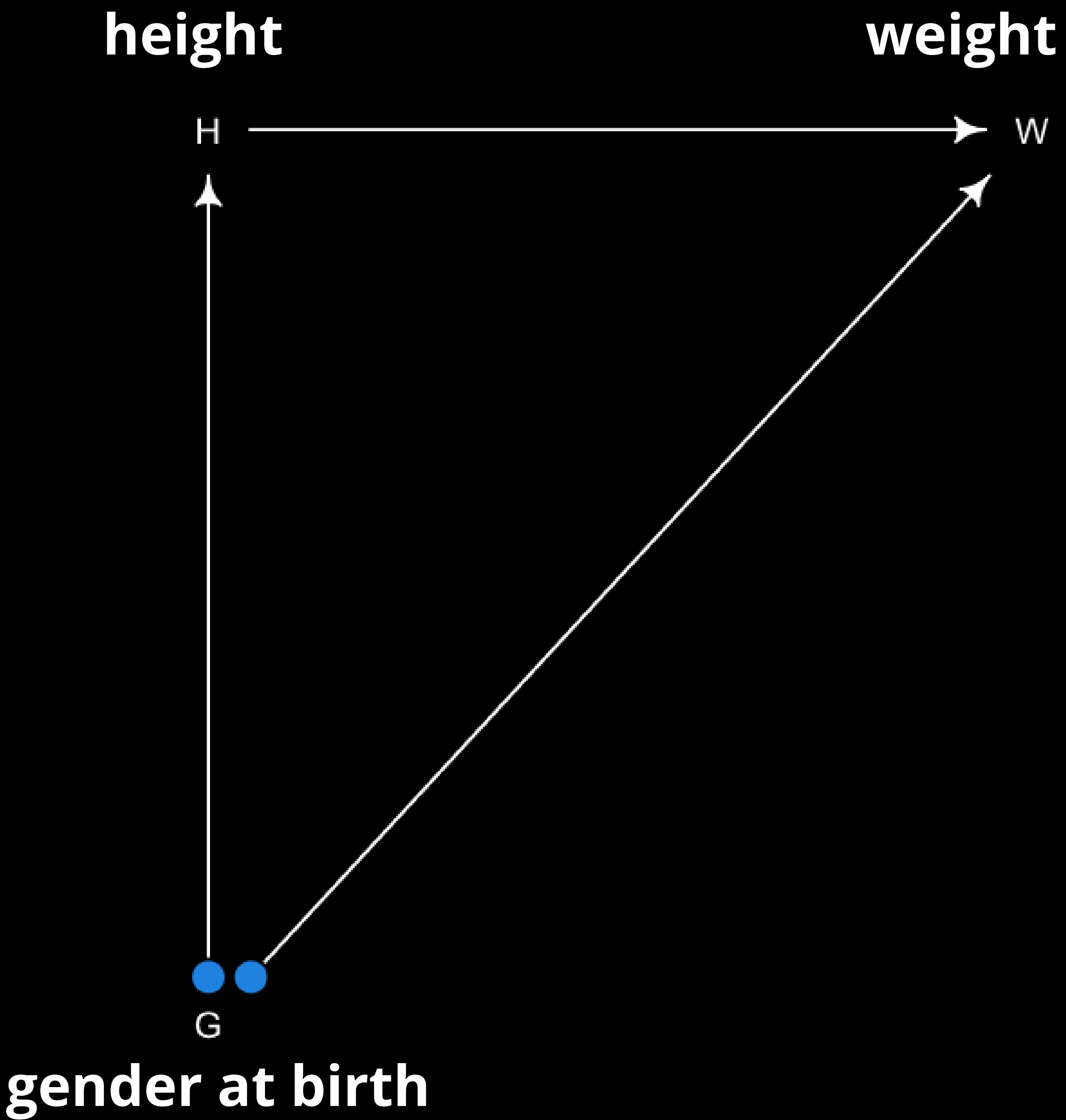
$$\begin{tikzcd} H \arrow[r] & W \\ G \arrow[u] \arrow[ur] & \end{tikzcd}$$

$$H = f_H(G)$$

$$
\begin{tikzcd}
H \arrow[r] & W \\
G \arrow[u] \arrow[ur] &
\end{tikzcd}
$$

$$H = f_H(G)$$

$$W = f_W(G, H)$$


height
weight
H
W
G
gender at birth

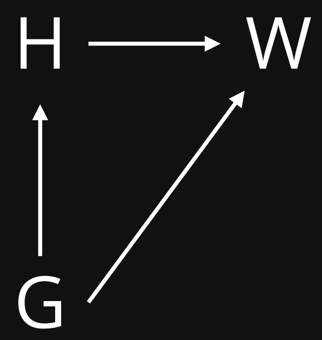
Think scientifically first

Different causal questions need different statistical models

Q: Causal effect of H on W?

Q: Total causal effect of G on W?

Q: Direct causal effect of G on W?



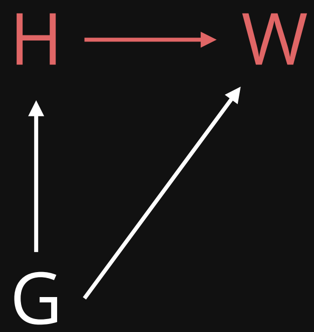
Think scientifically first

Different causal questions need different statistical models

Q: Causal effect of H on W?

Q: Total causal effect of G on W?

Q: Direct causal effect of G on W?



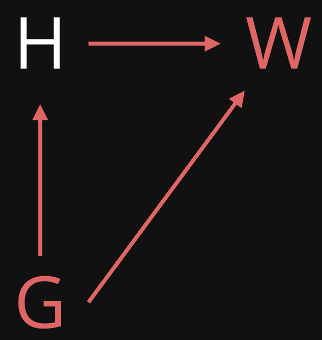
Think scientifically first

Different causal questions need different statistical models

Q: Causal effect of H on W?

Q: Total causal effect of G on W?

Q: Direct causal effect of G on W?



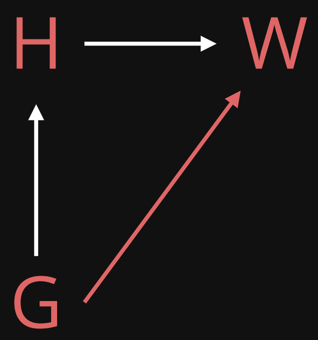
Think scientifically first

Different causal questions need different statistical models

Q: Causal effect of H on W?

Q: Total causal effect of G on W?

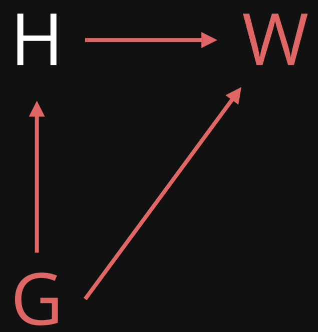
Q: Direct causal effect of G on W?



From estimand to estimate

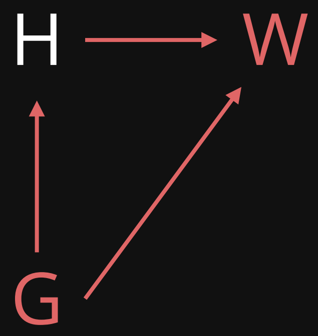
From estimand to estimate

Q: Total causal effect of G on W?

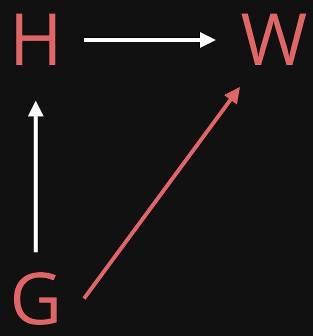


From estimand to estimate

Q: Total causal effect of G on W?

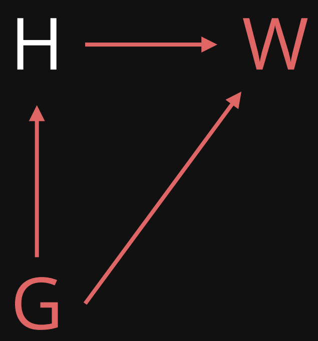


Q: Direct causal effect of G on W?

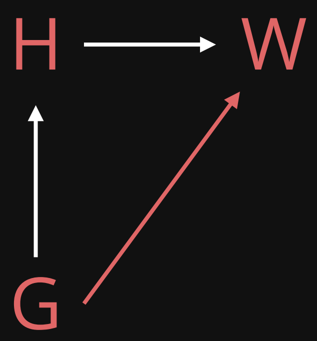


From estimand to estimate

Q: Total causal effect of G on W?



Q: Direct causal effect of G on W?



Need to model G as ***categorical*** variable

Working with categories

Several ways to code categorical variables

Working with categories

Several ways to code categorical variables

1. "dummy" (indicator) variables (0/1)

Working with categories

Several ways to code categorical variables

1. "dummy" (indicator) variables (0/1)

2. index variables: 0,1,2,3,...

Working with categories

Several ways to code categorical variables

1. "dummy" (indicator) variables (0/1)

2. index variables: 0,1,2,3,...

Working with categories

Several ways to code categorical variables

1. "dummy" (indicator) variables (0/1)

2. index variables: 0,1,2,3,...

extend to many categories without change in code

Working with categories

Several ways to code categorical variables

1. "dummy" (indicator) variables (0/1)

2. index variables: 0,1,2,3,...

extend to many categories without change in code

better for specifying priors

Working with categories

Several ways to code categorical variables

1. "dummy" (indicator) variables (0/1)

2. index variables: 0,1,2,3,...

extend to many categories without change in code

better for specifying priors

straight forward extension to multi-level models

Working with categories

Working with categories

Example: How does color influence t-shirt sales?

Working with categories

Example: How does color influence t-shirt sales?

Category	Cyan	Magenta	Yellow	Black
Index Value	0	1	2	3

Working with categories

Example: How does color influence t-shirt sales?

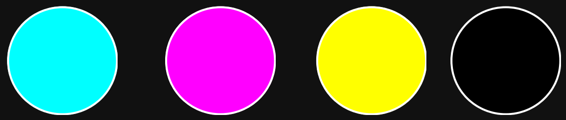
Category	Cyan	Magenta	Yellow	Black
Index Value	0	1	2	3

Influence of color coded by

$$\alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]$$

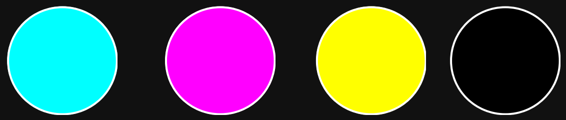
Working with categories

$$\alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]$$



Working with categories

$$\alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]$$

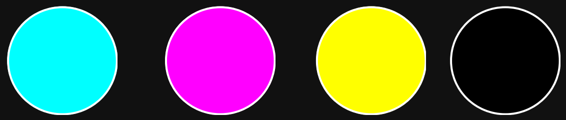


$$y_i \sim \mathrm{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{\mathrm{COLOR}[i]}$$

Working with categories

$$\alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]$$



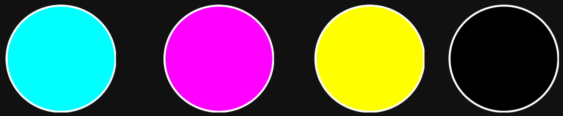
profits from sales of t-shirt *i*

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{\text{COLOR}[i]}$$

Working with categories

$$\alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]$$



profits from sales of t-shirt *i*

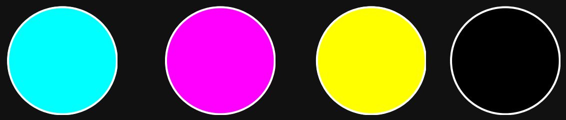
$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{\text{COLOR}[i]}$$

expected profits from
sales of t-shirt *i*

Working with categories

$$\alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]$$



profits from sales of t-shirt *i*

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{\text{COLOR}[i]}$$

color of t-shirt *i*

expected profits from
sales of t-shirt *i*

Using index variables

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha$$

intercept

Using index variables

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

Using index variables

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

gender of
i-th person

Using index variables

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

gender of *i*-th person

$G[i] = 0$ (female)

$G[i] = 1$ (male)

Using index variables

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

gender of *i*-th person

$G[i] = 0$ (female)

$G[i] = 1$ (male)

$$\alpha = [\alpha_0, \alpha_1]$$

two intercepts, one for each value of G

Using index variables

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$i = 0$

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{\boxed{G[i]}}$$

$G[0] = 1$

$$\alpha = [\alpha_0, \boxed{\alpha_1}]$$

Using index variables

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$i = 1$

$$W_i \sim \mathrm{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{\boxed{G[i]}}$$

$G[1] = 0$

$$\alpha = [\boxed{\alpha_0}, \alpha_1]$$

Using index variables

$$W_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

Using index variables

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

Priors

$$\alpha = [\alpha_0, \alpha_1] \qquad \alpha_j \sim \text{Normal}(60, 10)$$

$$j \in [0, 1]$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

$$\alpha_j \sim \mathrm{Normal}(60, 10)$$

$$\sigma \sim \mathrm{Uniform}(0, 10)$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

$$\boxed{\alpha_j \sim \text{Normal}(60, 10)}$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

need 2 alphas in model

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

$$\alpha_j \sim \text{Normal}(60, 10) \quad j \in [0, 1]$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\boxed{\mu_i = \alpha_{G[i]}}$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

pandas Series of 0/1 values

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

$$\alpha_j \sim \mathrm{Normal}(60, 10)$$

$$\sigma \sim \mathrm{Uniform}(0, 10)$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

pandas Series of 0/1 values

select the alpha value from vector based on value of **gen** for *i*th individual

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \underline{\alpha_{G[i]}}$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 4

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]}$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\boxed{\sigma \sim \text{Uniform}(0, 10)}$$

Using index variables

Code 4

```python
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male #gender for each individual

with pm.Model() as m_GW:
    a = pm.Normal('a', 60, 10, shape=2)
    mu = pm.Deterministic("mu", a[gen])
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GW_idata, _ = quap([a, sigma])
```

$$\boxed{W_i \sim \mathrm{Normal}(u_i, \sigma)}$$

$$\mu_i = \alpha_{G[i]}$$

$$\alpha_j \sim \mathrm{Normal}(60, 10)$$

$$\sigma \sim \mathrm{Uniform}(0, 10)$$

Posterior means & predictions

posterior mean weight (by gender)

Code 5

```
import arviz as az
import seaborn  as sns

post = az.extract(m_GW_idata, num_samples=1000)

# posterior mean weight
sns.kdeplot(post.a.values[1], color = "#e06666")
ax = sns.kdeplot(post.a.values[0], color = "cyan")
ax.set_xlabel("posterior mean weight (kg)")
sns.despine();
```

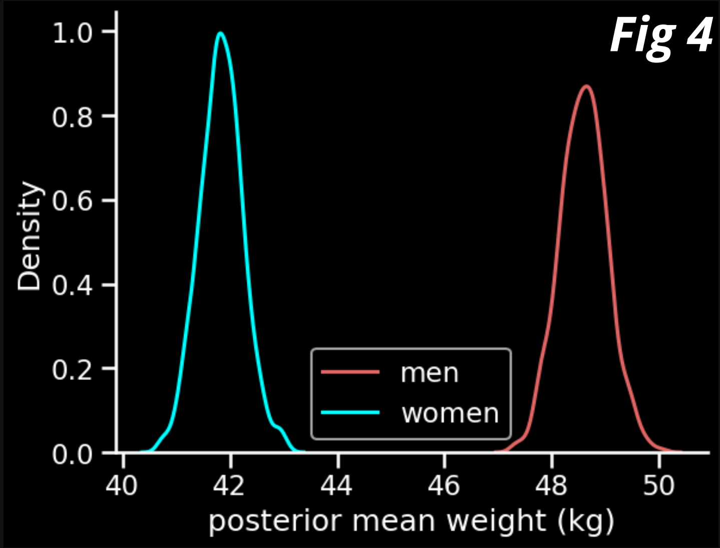
Posterior means & predictions

posterior mean weight (by gender)

Code 5

```
import arviz as az
import seaborn  as sns

post = az.extract(m_GW_idata, num_samples=1000)

# posterior mean weight
sns.kdeplot(post.a.values[1], color = "#e06666")
ax = sns.kdeplot(post.a.values[0], color = "cyan")
ax.set_xlabel("posterior mean weight (kg)")
sns.despine();
```

Fig 4



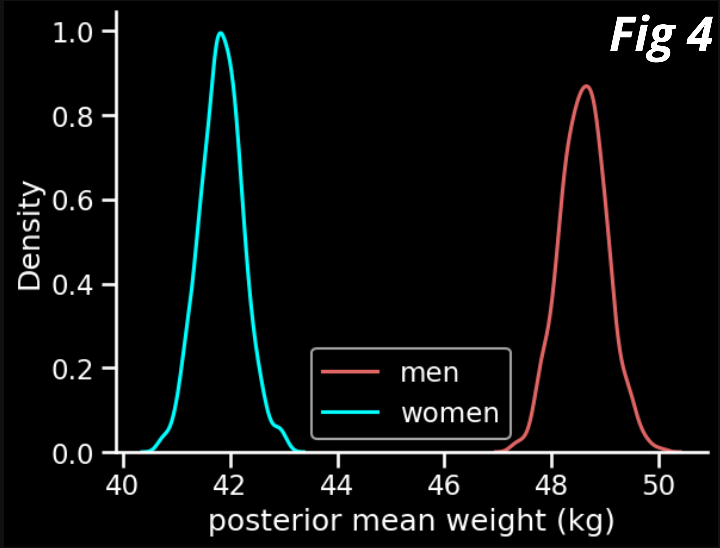
Posterior means & predictions

posterior mean weight (by gender)

Code 5

```python
import arviz as az
import seaborn  as sns

post = az.extract(m_GW_idata, num_samples=1000)

# posterior mean weight
sns.kdeplot(post.a.values[1], color = "#e06666")
ax = sns.kdeplot(post.a.values[0], color = "cyan")
ax.set_xlabel("posterior mean weight (kg)")
sns.despine();
```

Fig 4



posterior predicted weight (by gender)

Code 6

```python
w_W = stats.norm.rvs(post.a.values[0], post.sigma.values)
w_M = stats.norm.rvs(post.a.values[1], post.sigma.values)

_ = sns.kdeplot(w_W, color = "r")
ax = sns.kdeplot(w_M, color = "g")
_ = ax.set_xlabel("posterior predicted weight (kg)")
```

Posterior means & predictions

posterior mean weight (by gender)

Code 5

```
import arviz as az
import seaborn  as sns

post = az.extract(m_GW_idata, num_samples=1000)

# posterior mean weight
sns.kdeplot(post.a.values[1], color = "#e06666")
ax = sns.kdeplot(post.a.values[0], color = "cyan")
ax.set_xlabel("posterior mean weight (kg)")
sns.despine();
```

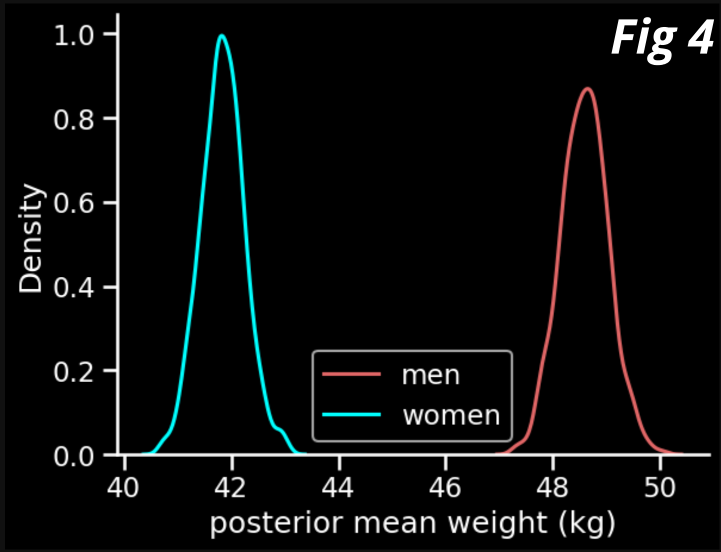


Fig 4

posterior predicted weight (by gender)

μ σ **Code 6**

```
w_W = stats.norm.rvs(post.a.values[0], post.sigma.values)
w_M = stats.norm.rvs(post.a.values[1], post.sigma.values)

_ = sns.kdeplot(w_W, color = "r")
ax = sns.kdeplot(w_M, color = "g")
_ = ax.set_xlabel("posterior predicted weight (kg)")
```

Posterior means & predictions

posterior mean weight (by gender)

Code 5

```
import arviz as az
import seaborn  as sns

post = az.extract(m_GW_idata, num_samples=1000)

# posterior mean weight
sns.kdeplot(post.a.values[1], color = "#e06666")
ax = sns.kdeplot(post.a.values[0], color = "cyan")
ax.set_xlabel("posterior mean weight (kg)")
sns.despine();
```

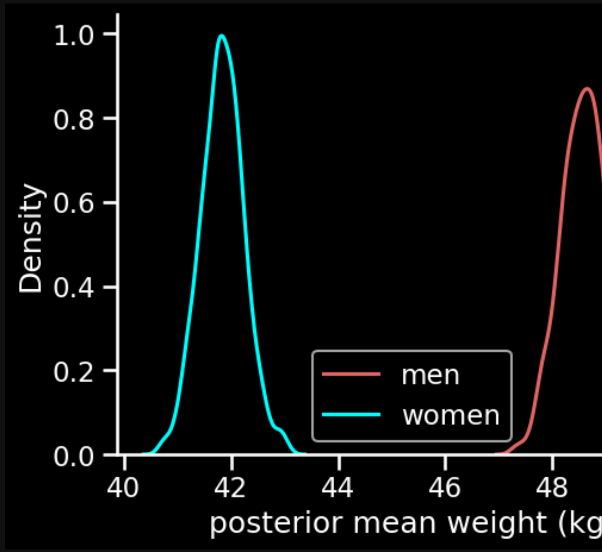


Fig 4

posterior predicted weight (by gender)

μ σ **Code 6**

```
w_W = stats.norm.rvs(post.a.values[0], post.sigma.values)
w_M = stats.norm.rvs(post.a.values[1], post.sigma.values)

_ = sns.kdeplot(w_W, color = "r")
ax = sns.kdeplot(w_M, color = "g")
_ = ax.set_xlabel("posterior predicted weight (kg)")
```

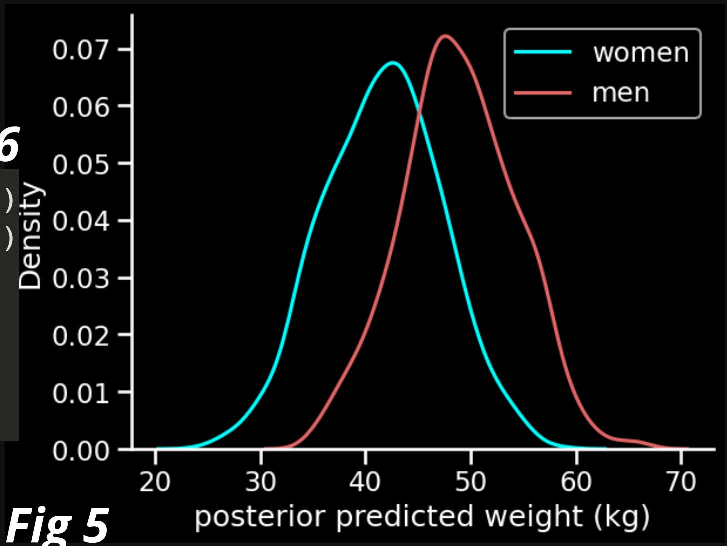


Fig 5

Always compute contrasts

Need to compute **contrast**

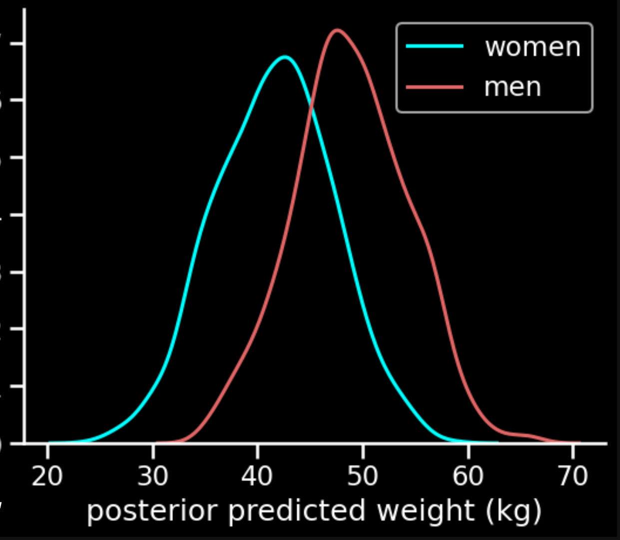


Fig 5

Always compute contrasts

Need to compute **contrast**
- difference between the categories

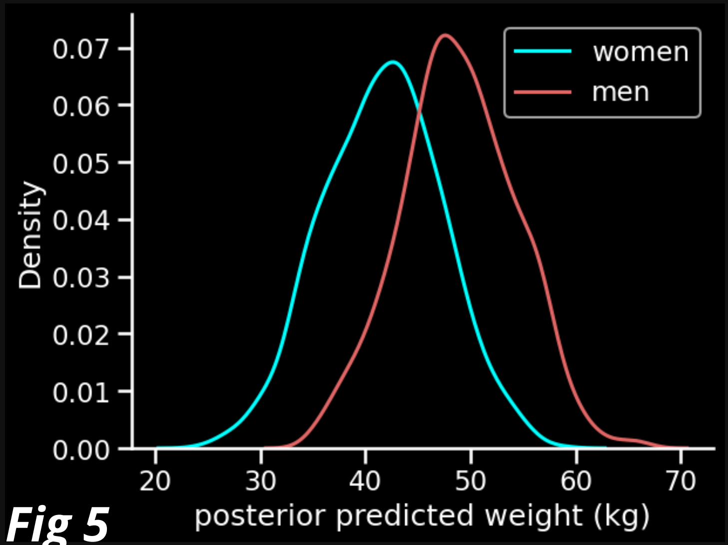


Fig 5

Always compute contrasts

Need to compute **contrast**
- difference between the categories

Don't want to simply compare **overlap** in parameters

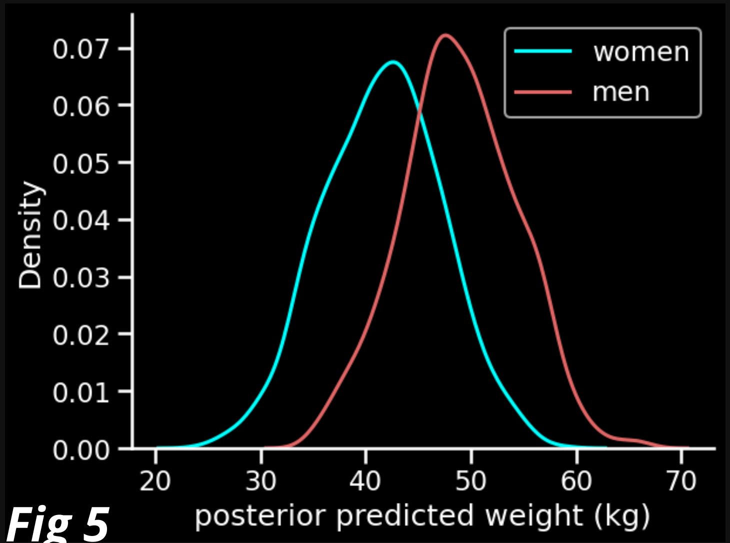


Fig 5

Always compute contrasts

Need to compute **contrast**
- difference between the categories

Don't want to simply compare **overlap** in parameters

Samples are not independent!

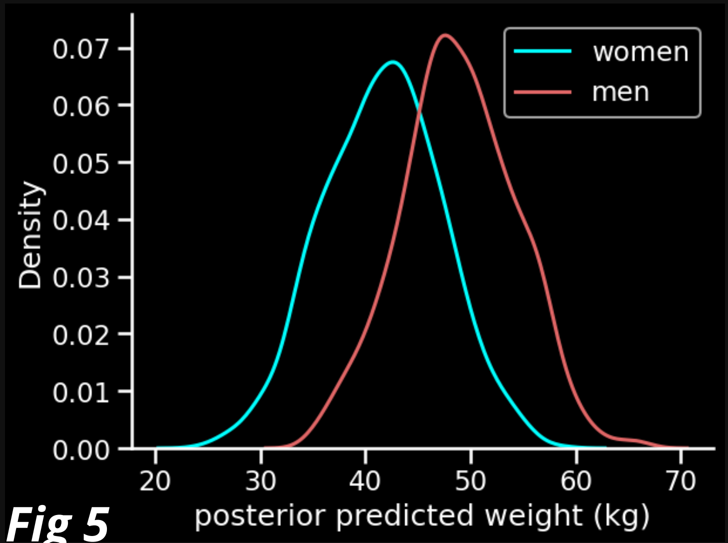


Fig 5

Always compute contrasts

Need to compute **contrast**
- difference between the categories

Don't want to simply compare **overlap** in parameters

Samples are not independent!

Compute **contrast distribution**

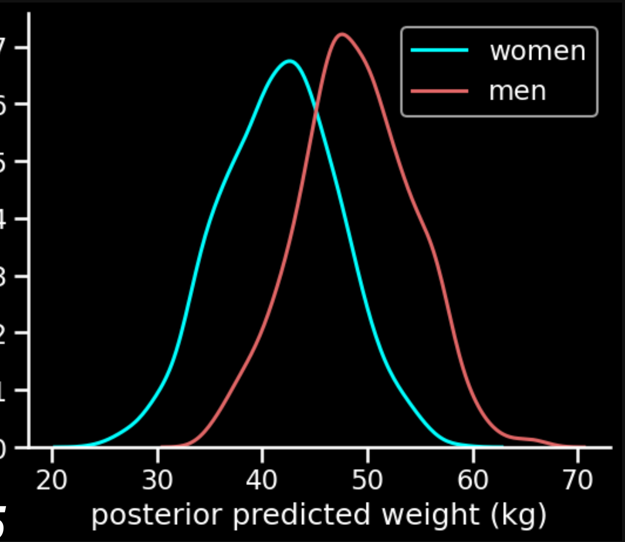
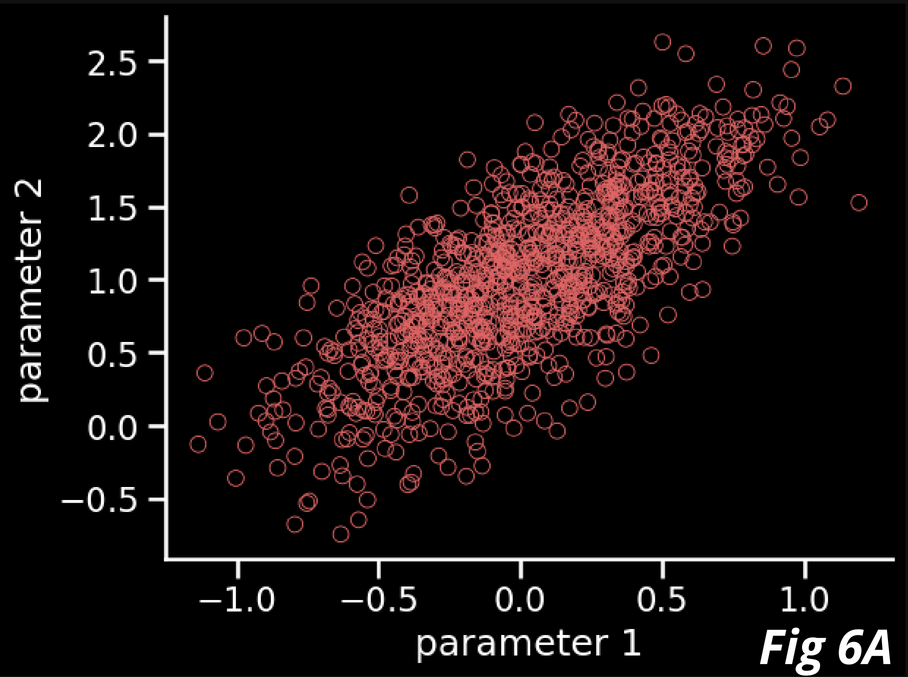


Fig 5

A motivation for computing contrasts



$p_{1,i} \sim \mathrm{Normal}(0, 0.4)$

$p_{2,i} \sim \mathrm{Normal}(p_{1,i} + 1, 0.4)$

$i \in [1, 1000]$

A motivation for computing contrasts

distributions of parameter values overlap

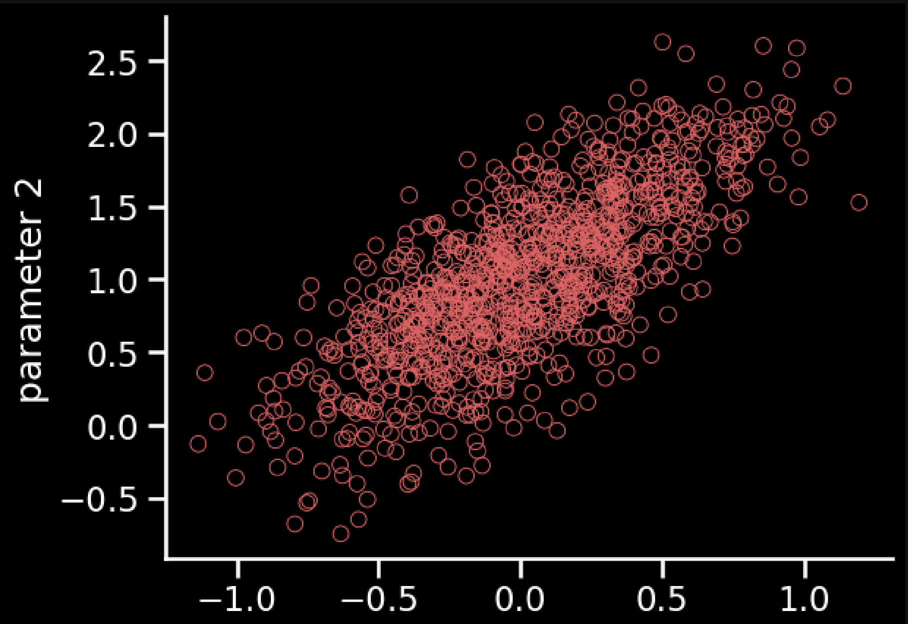


Fig 6A

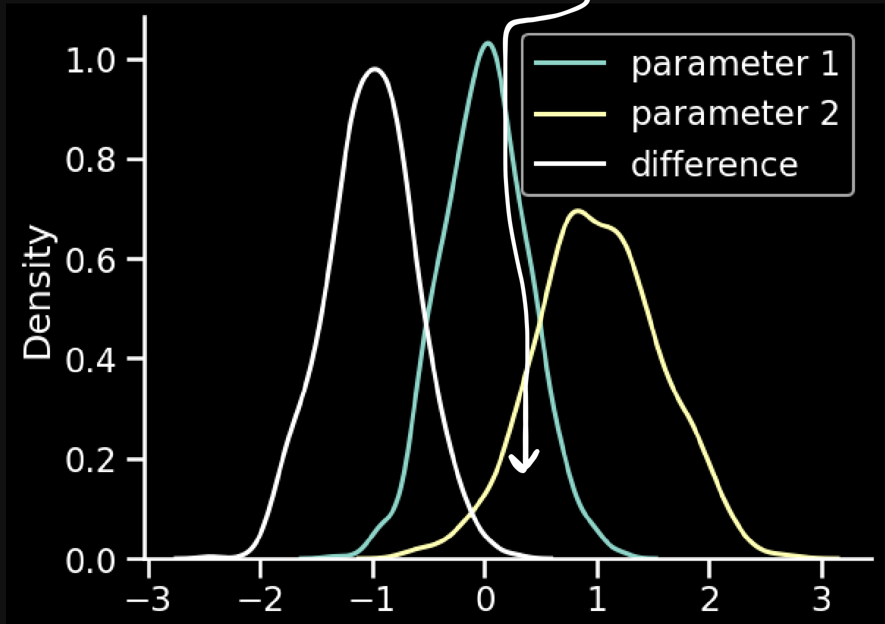


Fig 6B

$$p_{1,i} \sim \mathrm{Normal}(0, 0.4)$$

$$p_{2,i} \sim \mathrm{Normal}(p_{1,i} + 1, 0.4)$$

$$i \in [1, 1000]$$

A motivation for computing contrasts

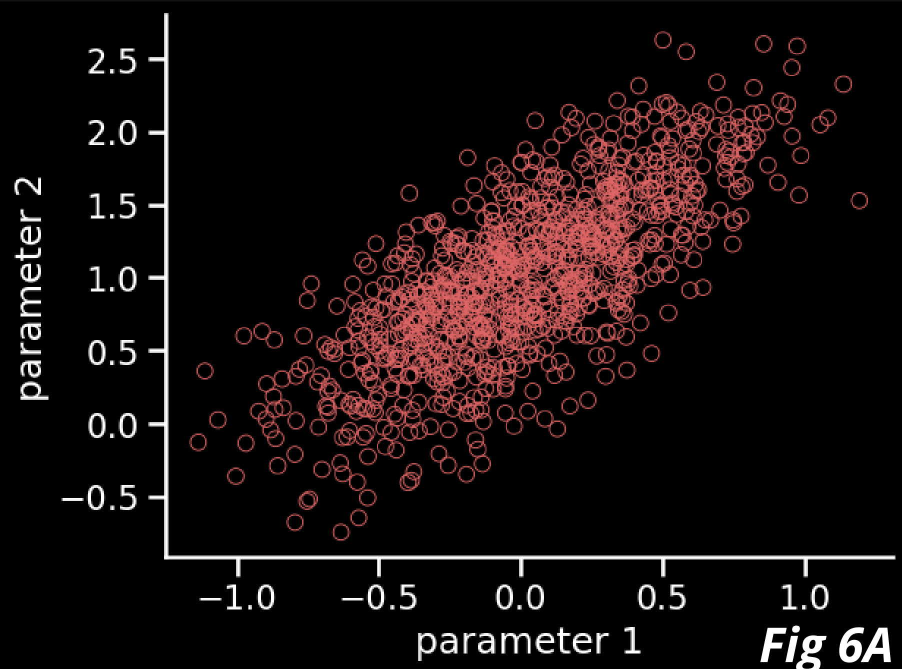


Fig 6A

distributions of parameter values overlap

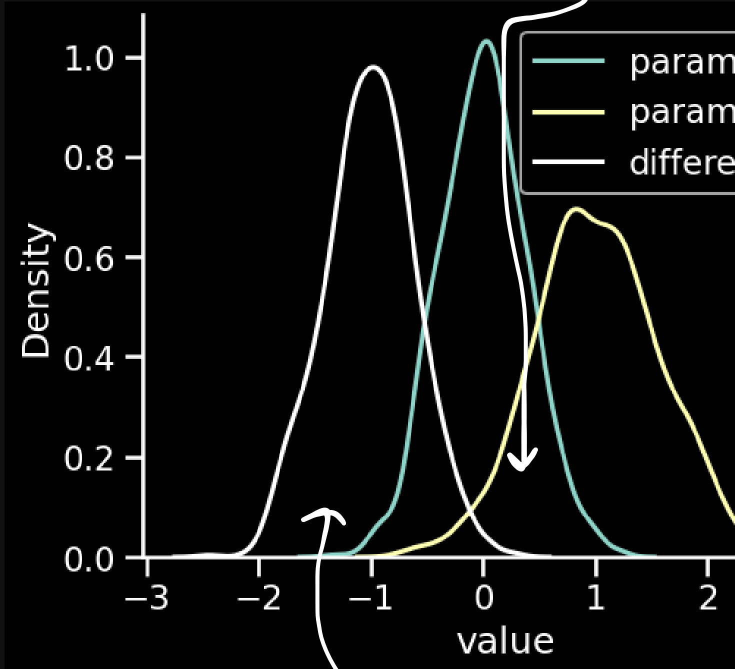


Fig 6B

contrast distribution shows non-zero difference between paired parameters

Causal Contrast

```
post
```

xarray.Dataset

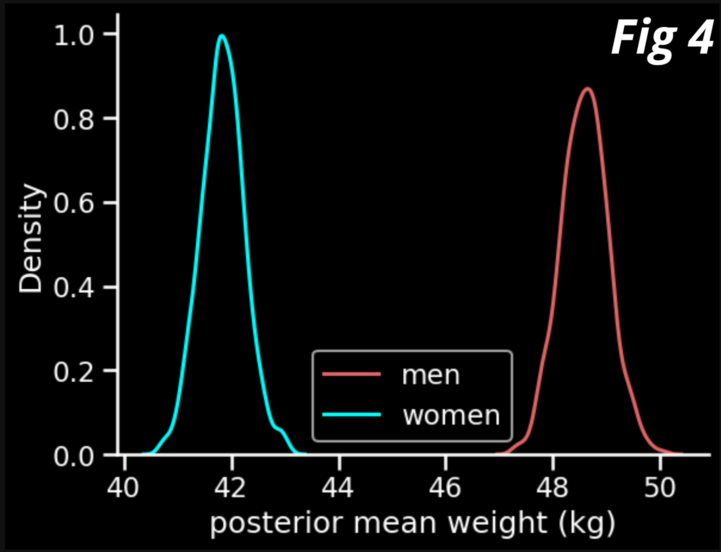
▶ Dimensions: (**a_dim_0**: 2, **sample**: 1000)

▼ Coordinates:

a_dim_0	(a_dim_0)	int64	0 1
sample	(sample)	MultiIndex	(chain, draw)
chain	(sample)	int64	0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0
draw	(sample)	int64	2597 3123 2861 ... 1921 4875 1934

▼ Data variables:

a	(a_dim_0, sample)	float64	41.79 42.22 41.84 ... 48.81 49.16
sigma	(sample)	float64	5.541 5.359 5.627 ... 5.483 5.095



Causal Contrast

```
post
```

xarray.Dataset

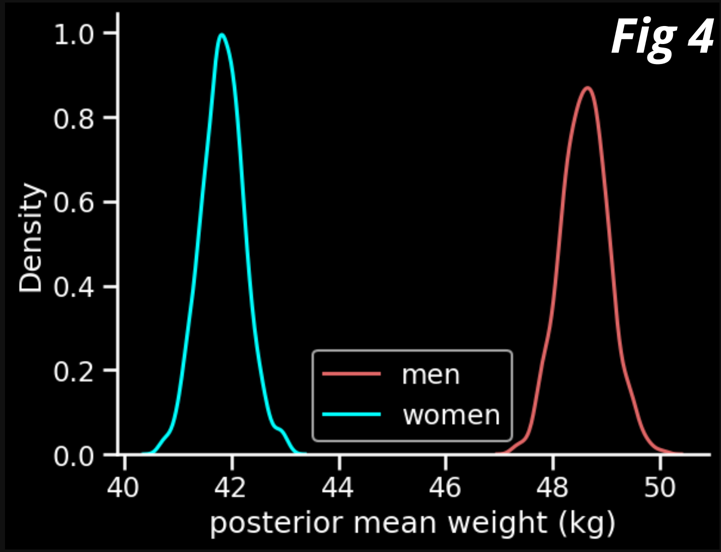
▸ **Dimensions:** (**a_dim_0**: 2, **sample**: 1000)

▾ **Coordinates:**

a_dim_0	(a_dim_0)	int64	0 1
sample	(sample)	MultiIndex	(chain, draw)
chain	(sample)	int64	0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0
draw	(sample)	int64	2597 3123 2861 ... 1921 4875 1934

▾ **Data variables:**

a	(a_dim_0, sample)	float64	41.79 42.22 41.84 ... 48.81 49.16
sigma	(sample)	float64	5.541 5.359 5.627 ... 5.483 5.095



Code 7

```python
import seaborn as sns

# causal contrast in means (male=1, female=0)
mu_contrast = post.a.values[1] - post.a.values[0]

ax = sns.kdeplot(mu_contrast, color = "w")
_ = ax.set_xlabel("posterior mean weight contrast (kg)")
```

Causal Contrast

```
post
```

xarray.Dataset

▶ Dimensions:	(**a_dim_0**: 2, **sample**: 1000)			
▼ Coordinates:				
a_dim_0	(a_dim_0)	int64	0 1	
sample	(sample)	MultiIndex	(chain, draw)	
chain	(sample)	int64	0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0	
draw	(sample)	int64	2597 3123 2861 ... 1921 4875 1934	
▼ Data variables:				
a	(a_dim_0, sample)	float64	41.79 42.22 41.84 ... 48.81 49.16	
sigma	(sample)	float64	5.541 5.359 5.627 ... 5.483 5.095	

Code 7

```python
import seaborn as sns

# causal contrast in means (male=1, female=0)
mu_contrast = post.a.values[1] - post.a.values[0]

ax = sns.kdeplot(mu_contrast, color = "w")
_ = ax.set_xlabel("posterior mean weight contrast (kg)")
```

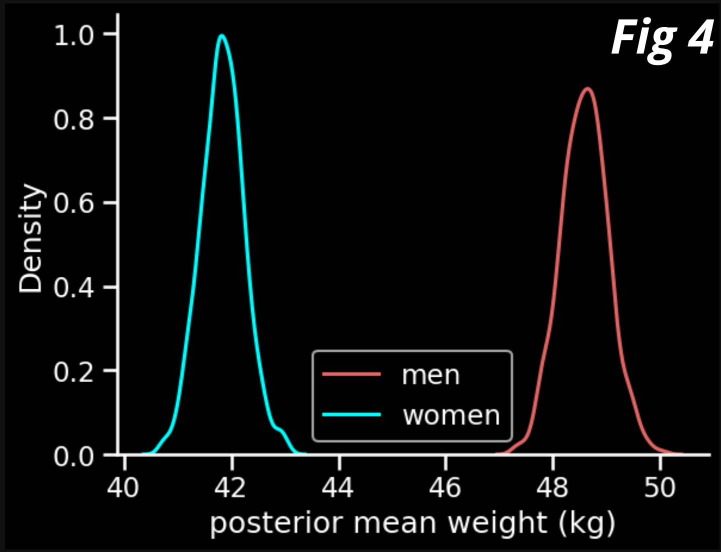


Fig 4

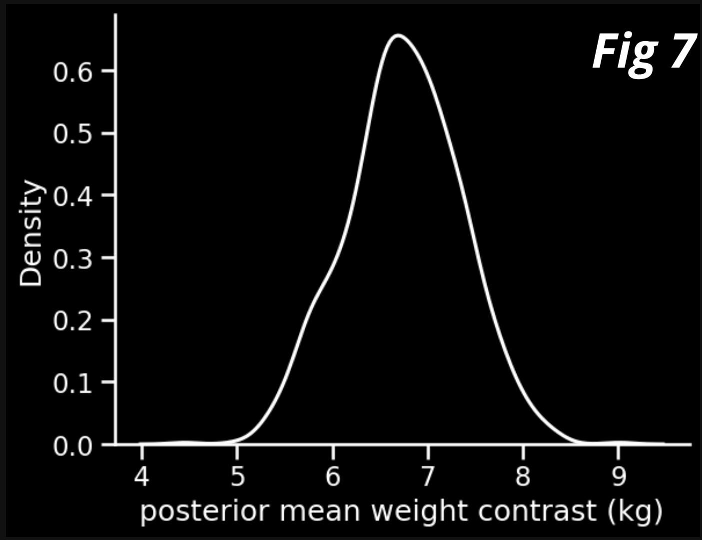


Fig 7

Causal Contrast

```
post
```

xarray.Dataset

▸ Dimensions: (**a_dim_0**: 2, **sample**: 1000)

▾ Coordinates:

a_dim_0	(a_dim_0)	int64	0 1
sample	(sample)	MultiIndex	(chain, draw)
chain	(sample)	int64	0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0
draw	(sample)	int64	2597 3123 2861 ... 1921 4875 1934

▾ Data variables:

a	(a_dim_0, sample)	float64	41.79 42.22 41.84 ... 48.81 49.16
sigma	(sample)	float64	5.541 5.359 5.627 ... 5.483 5.095

Code 7

```python
import seaborn as sns

# causal contrast in means (male=1, female=0)
mu_contrast = post.a.values[1] - post.a.values[0]

ax = sns.kdeplot(mu_contrast, color = "w")
_ = ax.set_xlabel("posterior mean weight contrast (kg)")
```

reliably positive difference for men vs. women (on average)

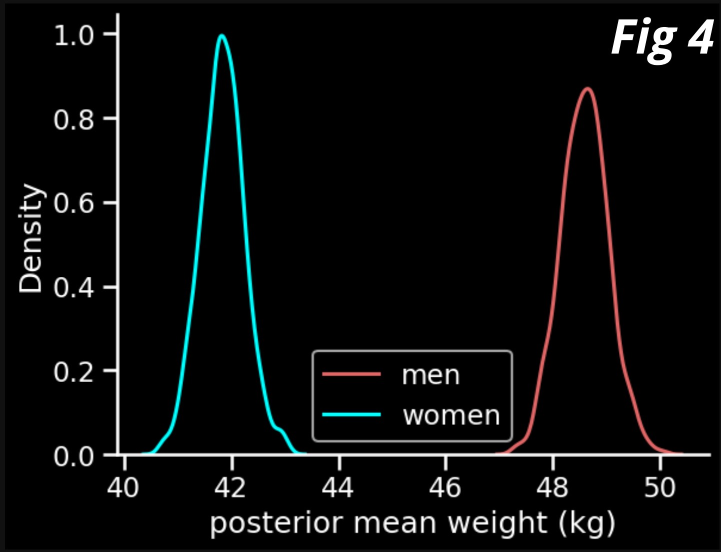


Fig 4

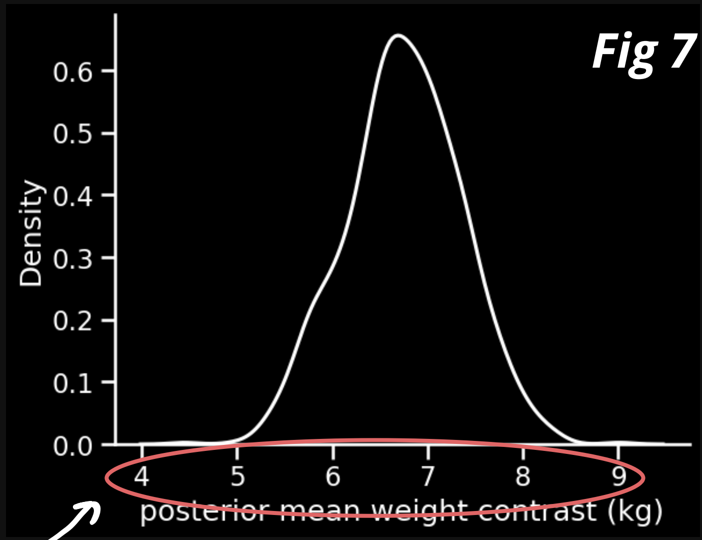


Fig 7

Simulated Weight Contrast

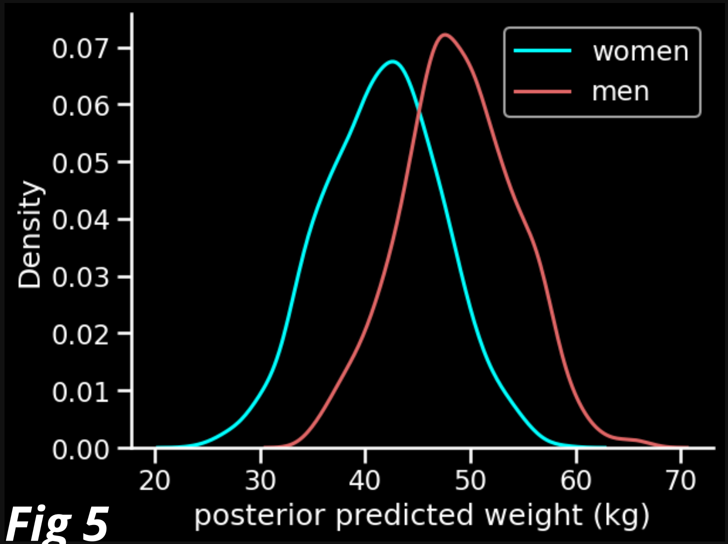


Fig 5

Simulated Weight Contrast

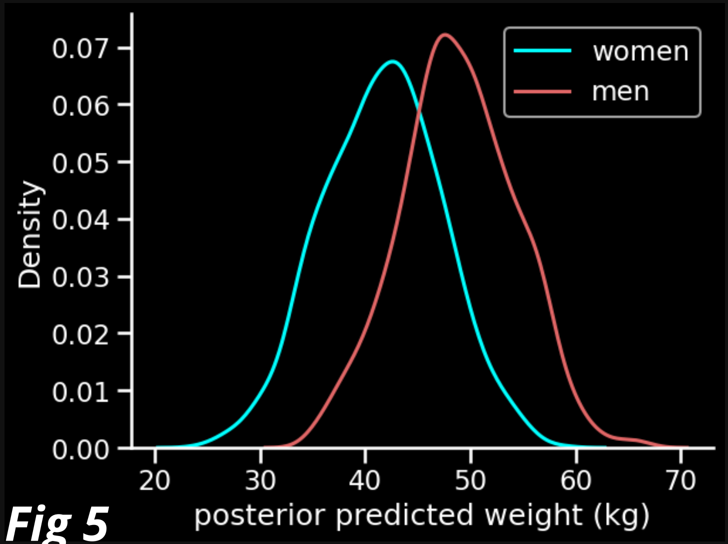


Fig 5

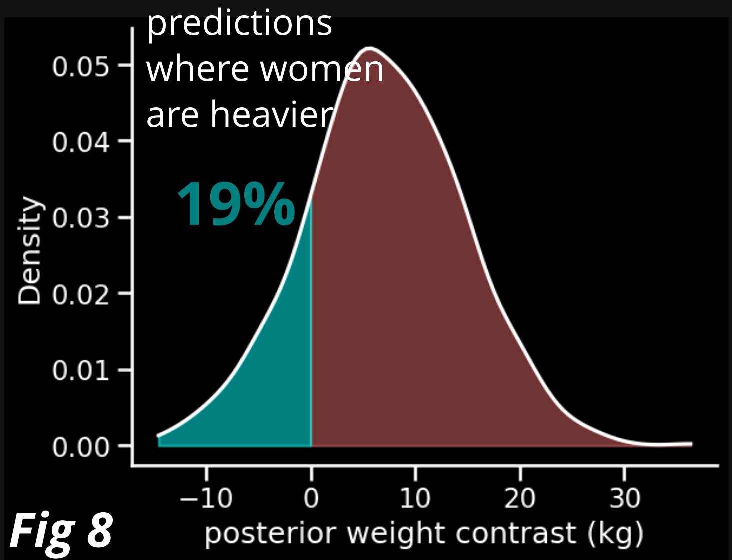


Fig 8

Simulated Weight Contrast

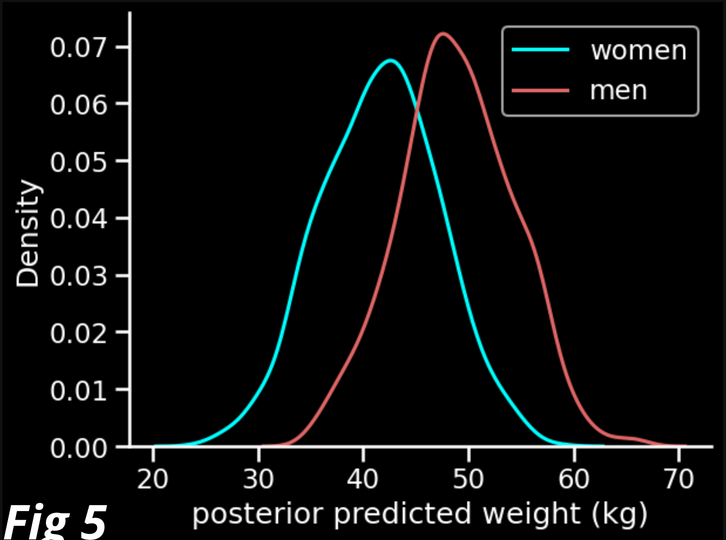


Fig 5

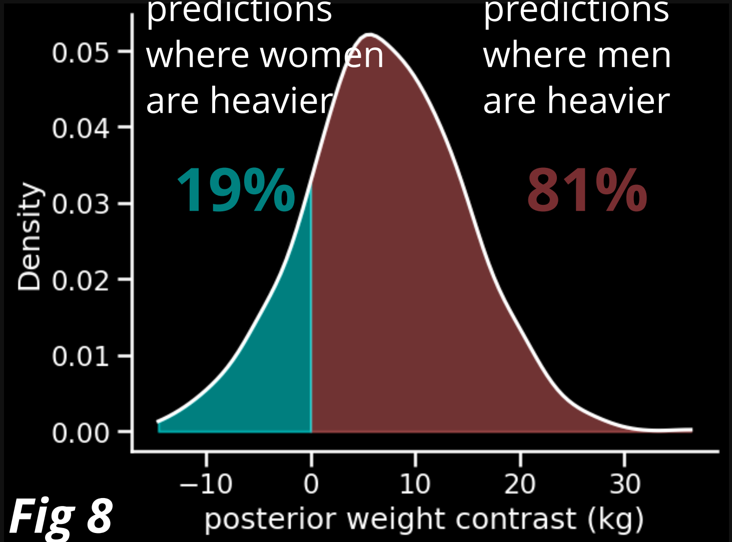
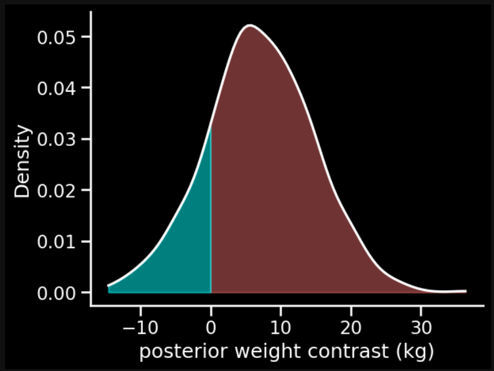
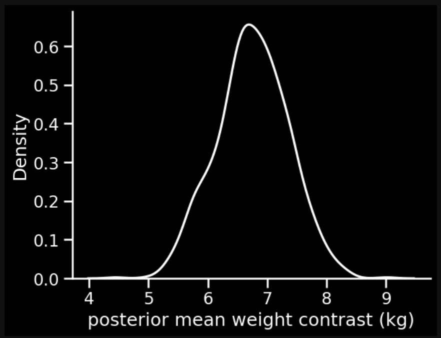


Fig 8

From estimand to estimate

Q: Total causal effect of G on W?



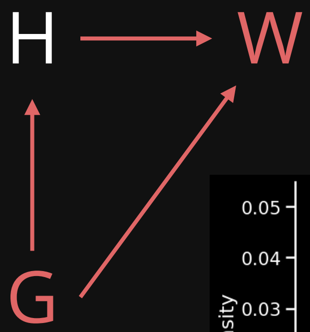


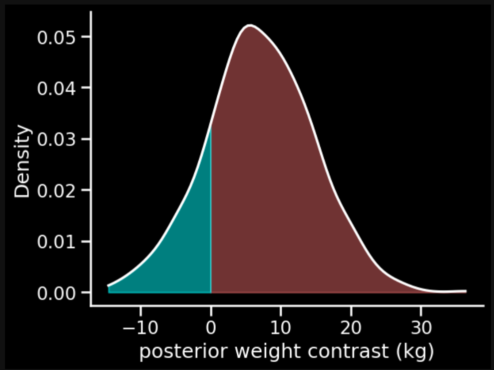
From estimand to estimate

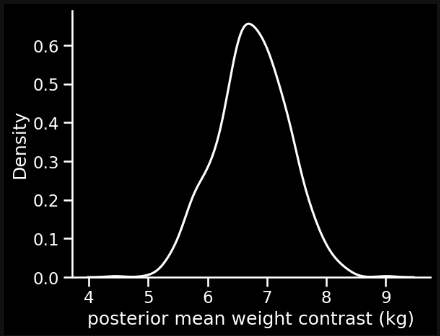
influence of G on W

- direct influence
- AND influence through H

Q: <u>Total causal effect</u> of G on W?

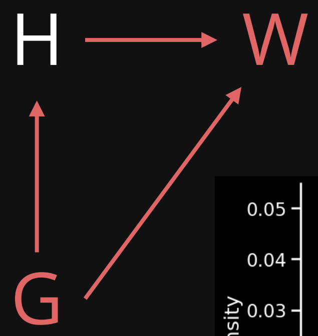


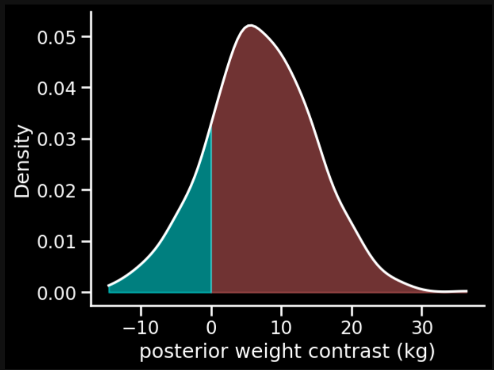


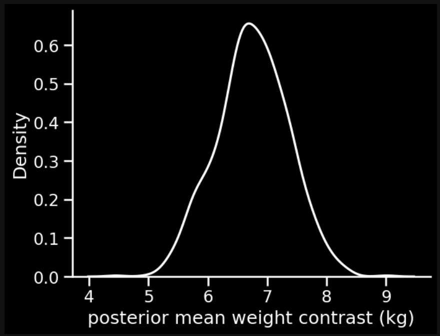


From estimand to estimate

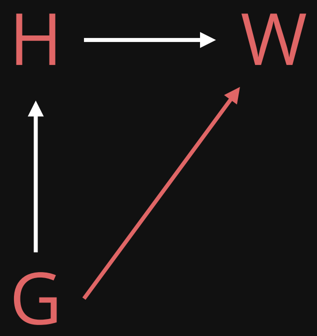
Q: Total causal effect of G on W?





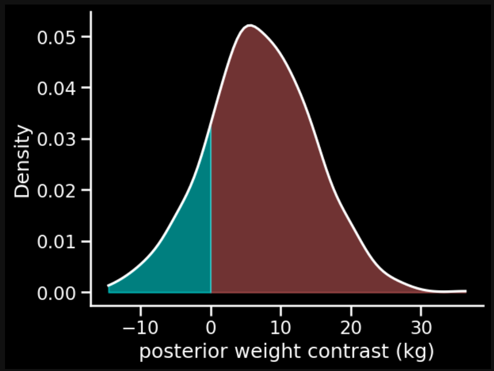


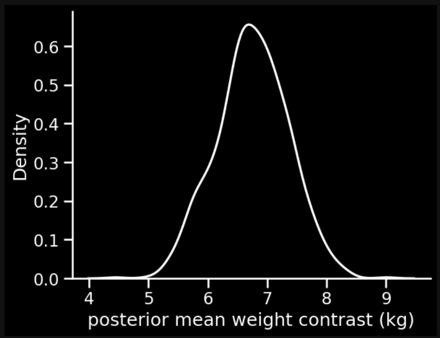
Q: Direct causal effect of G on W?



From estimand to estimate

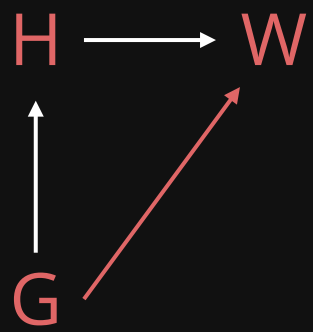
Q: Total causal effect of G on W?





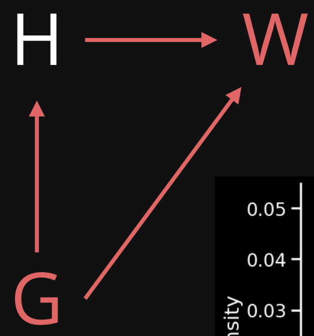
direct influence of G on W (ONLY)

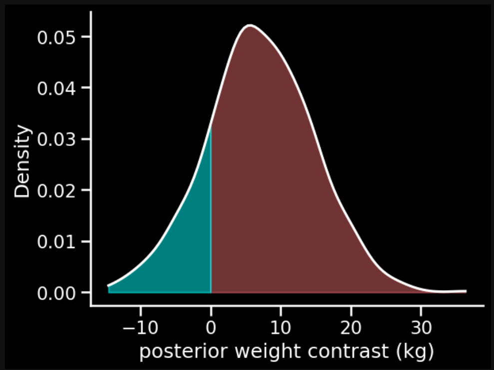
Q: Direct causal effect of G on W?

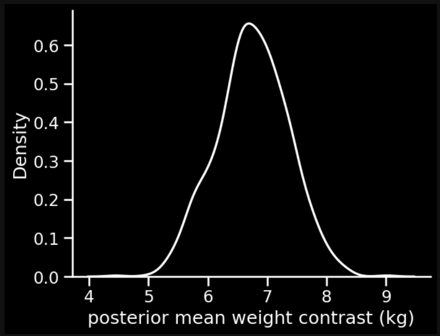


From estimand to estimate

Q: Total causal effect of G on W?

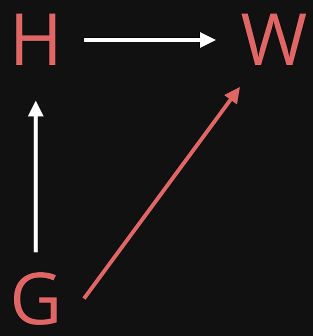






direct influence of G on W (ONLY)

Q: Direct causal effect of G on W?



need a different statistical model for estimate...

Index variables and lines

model without stratification by gender

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha + \beta(H_i - \bar{H})$$

Index variables and lines

model without stratification by gender

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha + \beta(H_i - \bar{H})$$

intercept

slope

Index variables and lines

model with stratification by gender

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

model with stratification by gender

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

model with stratification by gender

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

gender of
i-th person

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

model with stratification by gender

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

gender of i-th person

$G[i] = 0$ (female)

$G[i] = 1$ (male)

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

model with stratification by gender

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

gender of i-th person

$G[i] = 0$ (female)

$G[i] = 1$ (male)

$$\alpha = [\alpha_0, \alpha_1]$$

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

model with stratification by gender

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

gender of i-th person

$G[i] = 0$ (female)

$G[i] = 1$ (male)

$$\alpha = [\alpha_0, \alpha_1] \qquad \beta = [\beta_0, \beta_1]$$

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

model with stratification by gender

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

gender of i-th person

$G[i] = 0$ (female)

$G[i] = 1$ (male)

$$\alpha = [\alpha_0, \alpha_1] \qquad \beta = [\beta_0, \beta_1]$$

Two intercepts and two slopes - one for each value in G

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$i = 0$

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{\boxed{G[i]}} + \beta_{\boxed{G[i]}}(H_i - \bar{H})$$

$G[0] = 1$

$$\alpha = [\alpha_0, \boxed{\alpha_1}] \qquad \beta = [\beta_0, \boxed{\beta_1}]$$

Index variables and lines

	H	W	G
0	152	48	1
1	140	36	0
2	137	32	0
3	157	53	1
4	141	45	0
5	164	63	1
6	149	38	0
7	169	55	1
8	148	35	0
9	165	54	1
10	154	50	0
11	151	41	1

$i = 1$

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{\boxed{G[i]}} + \beta_{\boxed{G[i]}}(H_i - \bar{H})$$

$G[1] = 0$

$$\alpha = [\boxed{\alpha_0}, \alpha_1] \qquad \beta = [\boxed{\beta_0}, \beta_1]$$

Using index variables

Code 8

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\beta_j \sim \text{LogNormal}(0, 2)$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 8

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\boxed{\alpha_j \sim \text{Normal}(60, 10)}$$

$$\beta_j \sim \text{LogNormal}(0, 2)$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 8

```python
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \mathrm{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\alpha_j \sim \mathrm{Normal}(60, 10)$$

$$\beta_j \sim \mathrm{LogNormal}(0, 2)$$

$$\sigma \sim \mathrm{Uniform}(0, 10)$$

Using index variables

Code 8

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\boxed{\beta_j \sim \text{LogNormal}(0, 2)}$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 8

```python
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\beta_j \sim \text{LogNormal}(0, 2)$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 8

```python
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform('sigma', 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\boxed{\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})}$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\beta_j \sim \text{LogNormal}(0, 2)$$

$$\sigma \sim \text{Uniform}(0, 10)$$

Using index variables

Code 8

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$W_i \sim \text{Normal}(u_i, \sigma)$$

$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\beta_j \sim \text{LogNormal}(0, 2)$$

$$\boxed{\sigma \sim \text{Uniform}(0, 10)}$$

Using index variables

Code 8

```
import pandas as pd
import pymc as pm
from quap import quap

df = pd.read_csv("Data/Howell1.csv", sep=';', header=0)
df2 = df[df.age >= 18]
gen = df2.male

with pm.Model() as m_GHW:
    a = pm.Normal('a', 60, 10, shape=2)
    b = pm.Lognormal('b', 0, 2, shape=2)
    mu = pm.Deterministic("mu", a[gen] + b[gen] * (df2.height - df2.height.mean()))
    sigma = pm.Uniform("sigma", 0, 10)

    weight = pm.Normal("weight", mu, sigma, observed=df2.weight)

    m_GHW_idata, _ = quap([a, b, sigma])
```

$$\boxed{W_i \sim \text{Normal}(u_i, \sigma)}$$

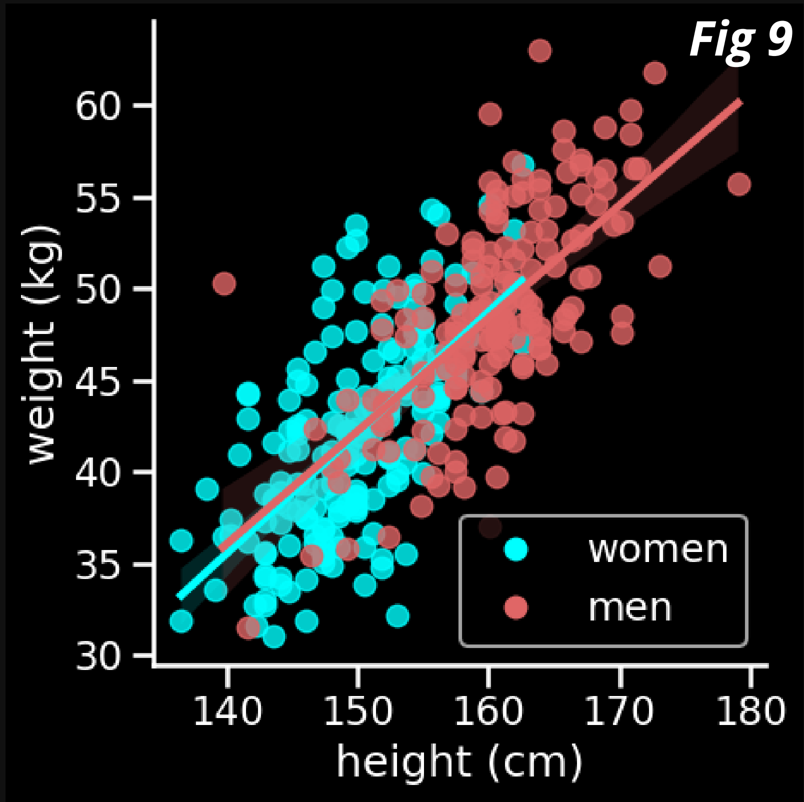
$$\mu_i = \alpha_{G[i]} + \beta_{G[i]}(H_i - \bar{H})$$

$$\alpha_j \sim \text{Normal}(60, 10)$$

$$\beta_j \sim \text{LogNormal}(0, 2)$$

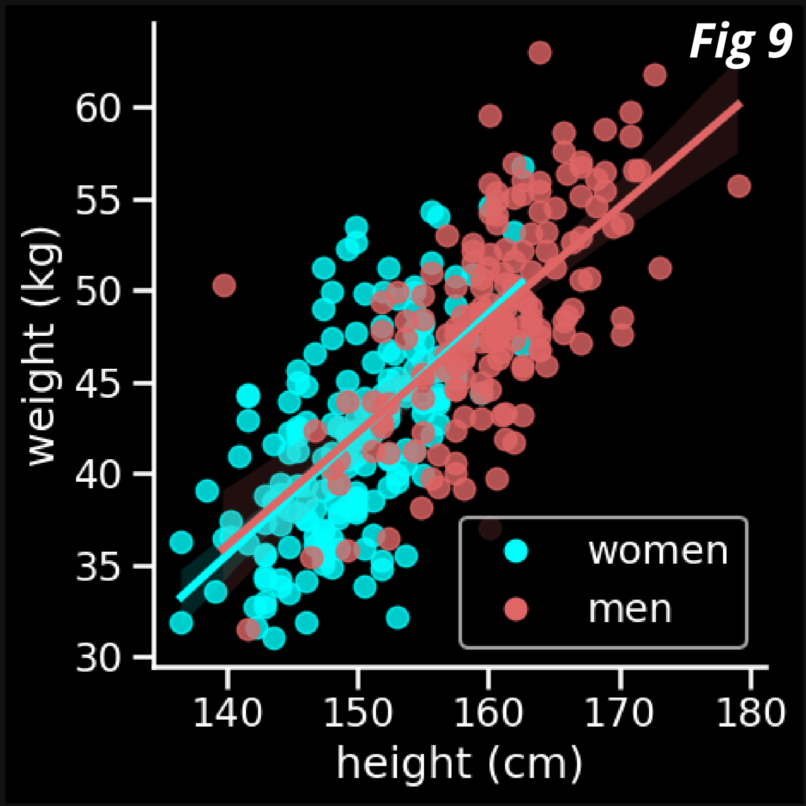
$$\sigma \sim \text{Uniform}(0, 10)$$

Contrasts at each height



Contrasts at each height

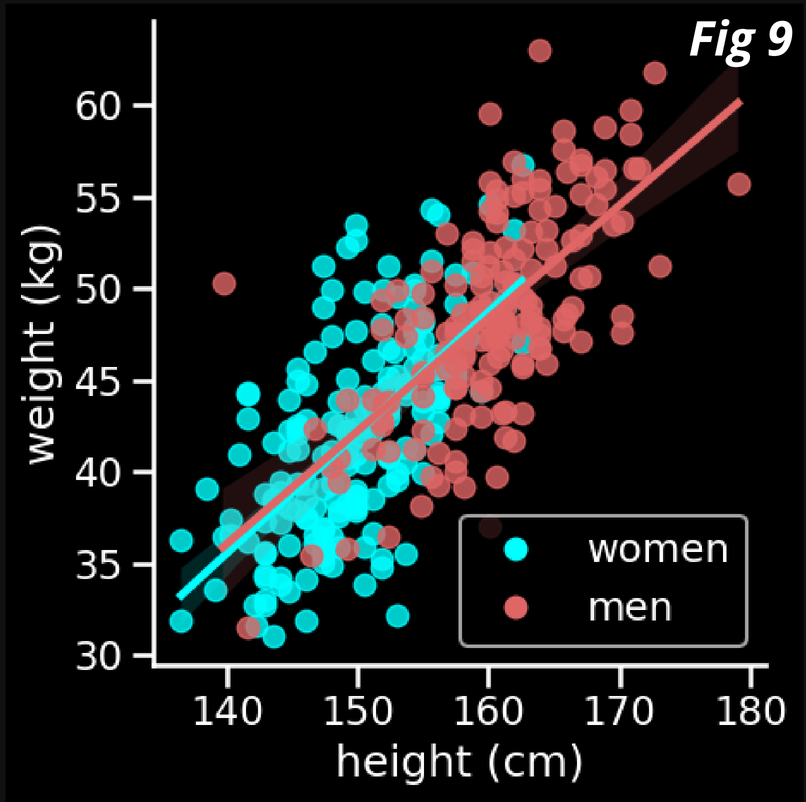
Don't rely on a plot such as this one for determining differences in influence of gender on weight



Contrasts at each height

(1) Compute posterior mean weight for women

Don't rely on a plot such as this one for determining differences in influence of gender on weight

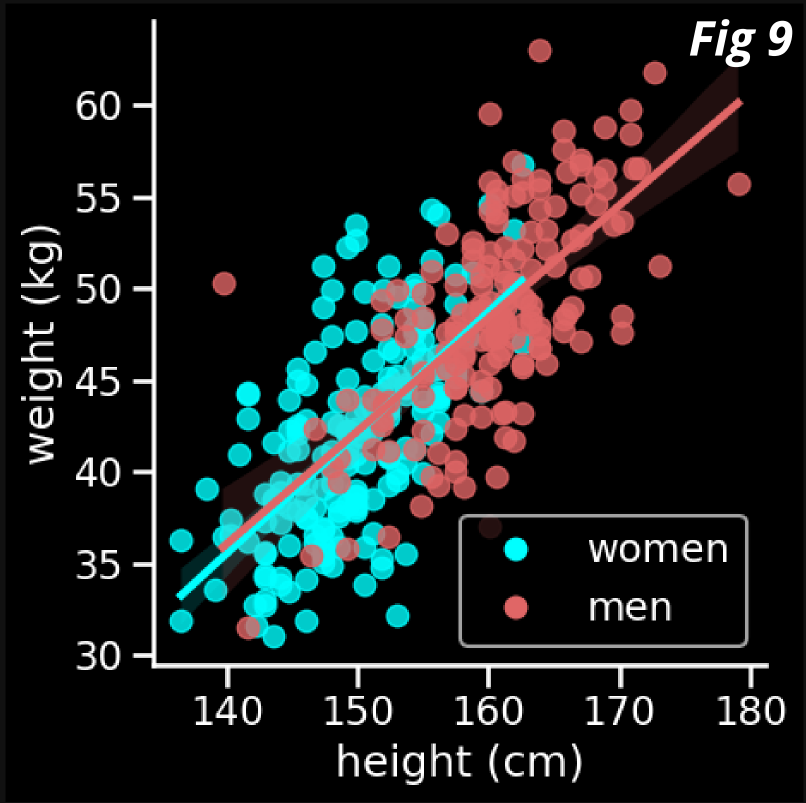


Contrasts at each height

(1) Compute posterior mean weight for women

(2) Compute posterior mean weight for men

Don't rely on a plot such as this one for determining differences in influence of gender on weight



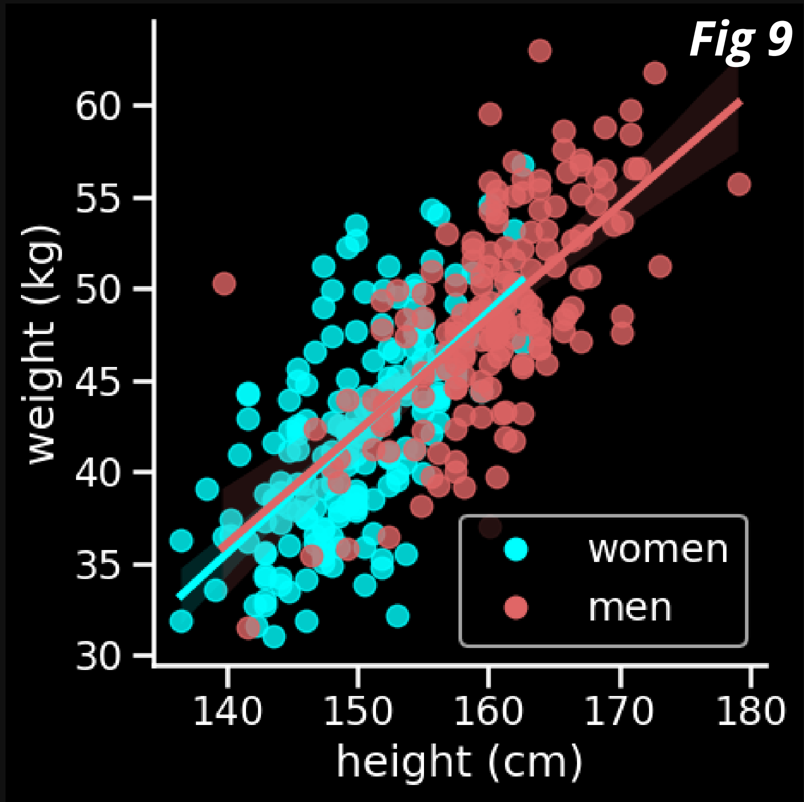
Contrasts at each height

(1) Compute posterior mean weight for women

(2) Compute posterior mean weight for men

(3) Subtract (2) from (1)

Don't rely on a plot such as this one for determining differences in influence of gender on weight



Contrasts at each height

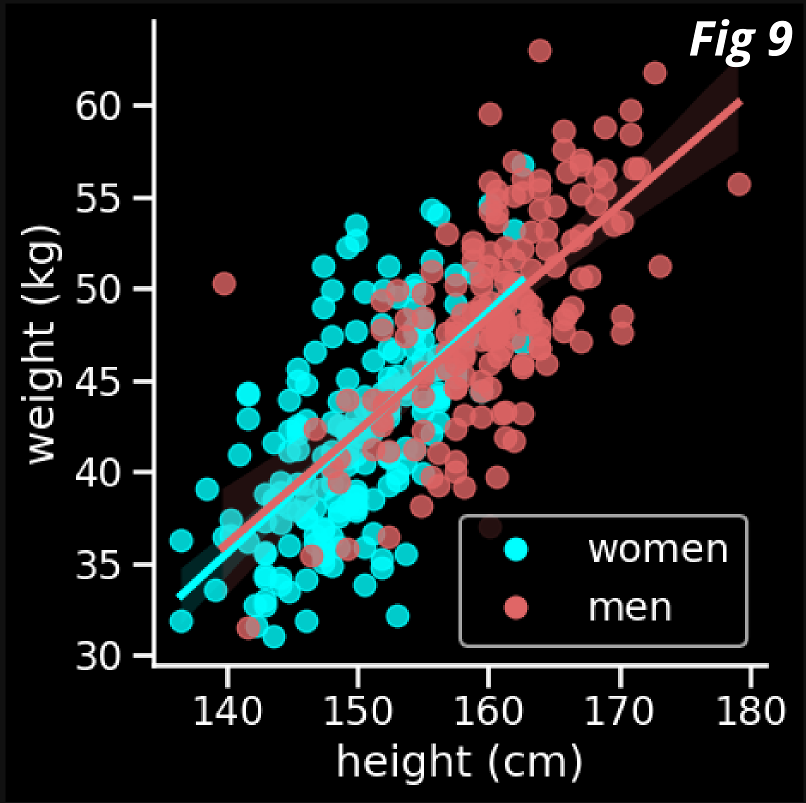
(1) Compute posterior mean weight for women

(2) Compute posterior mean weight for men

(3) Subtract (2) from (1)

(4) Plot contrast distribution at each height

Don't rely on a plot such as this one for determining differences in influence of gender on weight



Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
_ = sns.despine()
```

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)
```

grid of heights to use
for predictions

```
data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
_ = sns.despine()
```

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
_ = sns.despine()
```

extract 1000 samples
from posterior

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
_ = sns.despine()
```

storage for posterior mean weights by gender

Contrasts at each height

Code 9

```python
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
_ = sns.despine()
```

calculate 1000 expected weights for each height from posterior samples (by gender)

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

compute contrast distribution
for expected weights

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

vary high density interval size

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

plot high density interval for contrast distribution

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

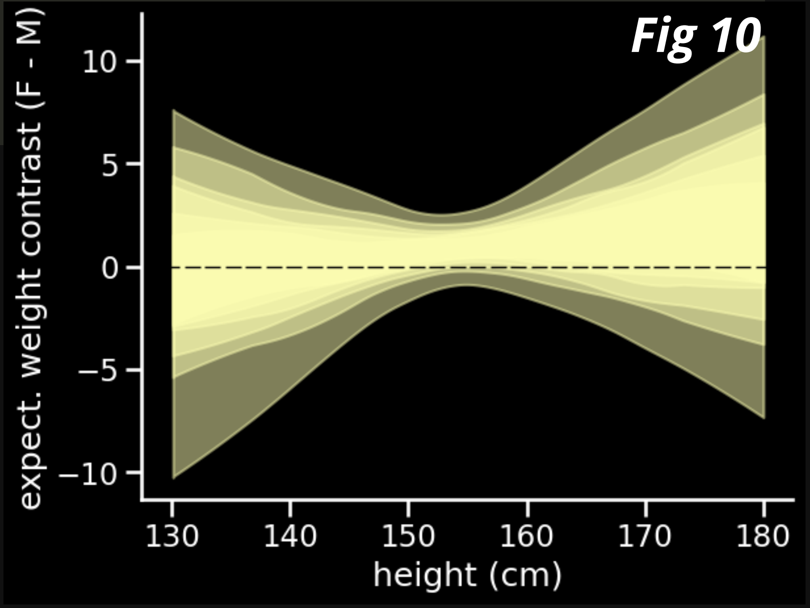
plot line representing NO difference in expected weights

Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

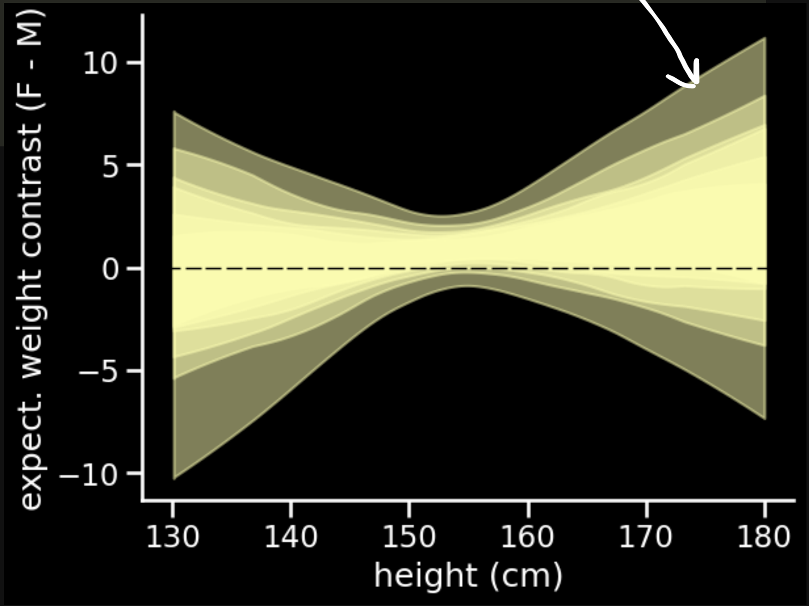
Fig 10



Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

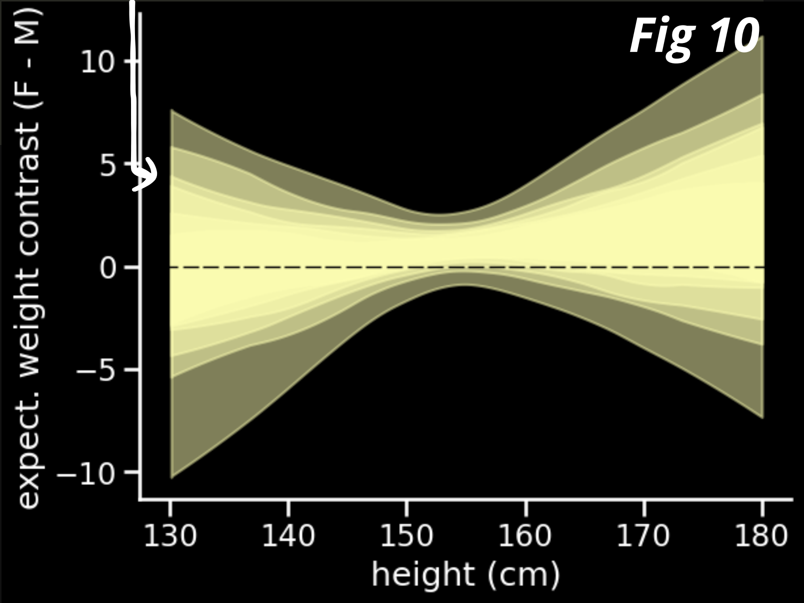


Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

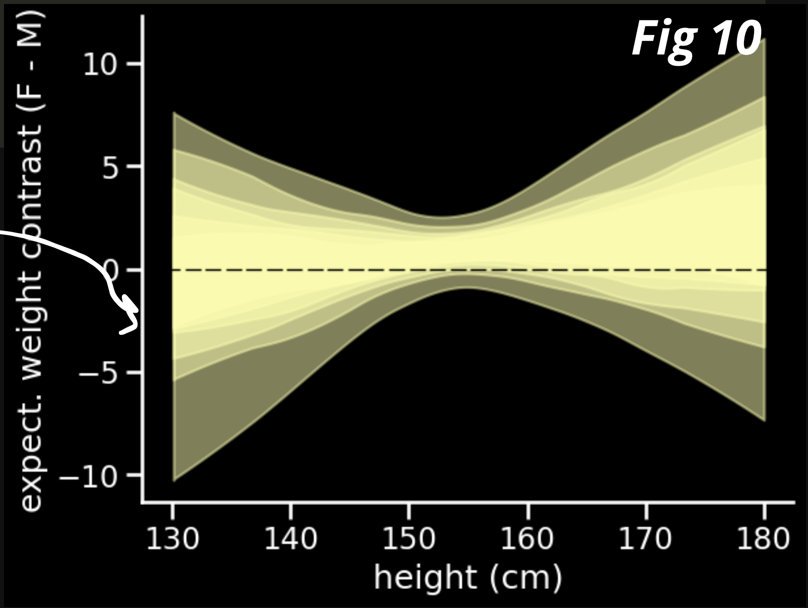
Fig 10



Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

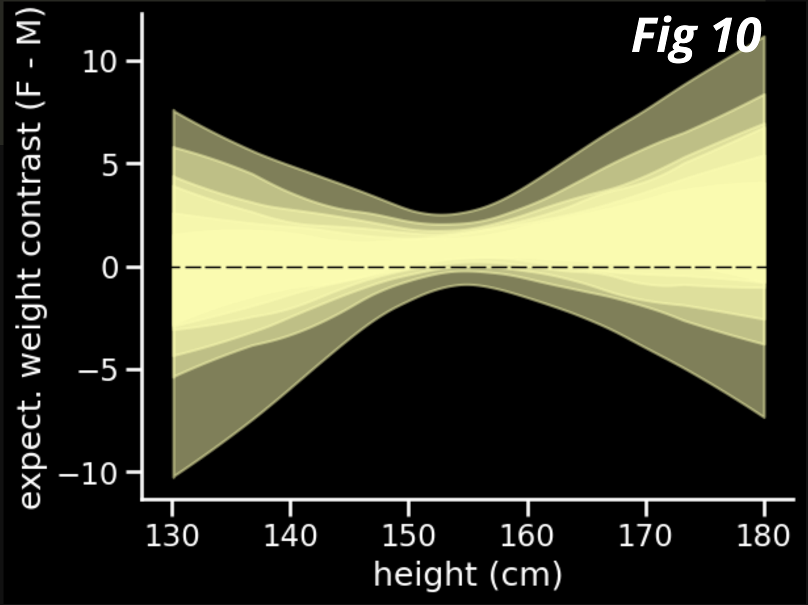


Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

Fig 10

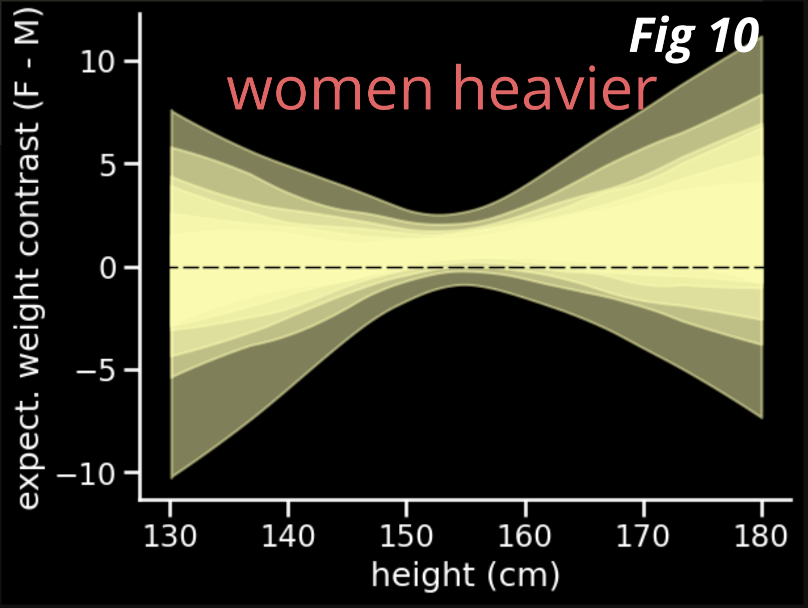


Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

Fig 10

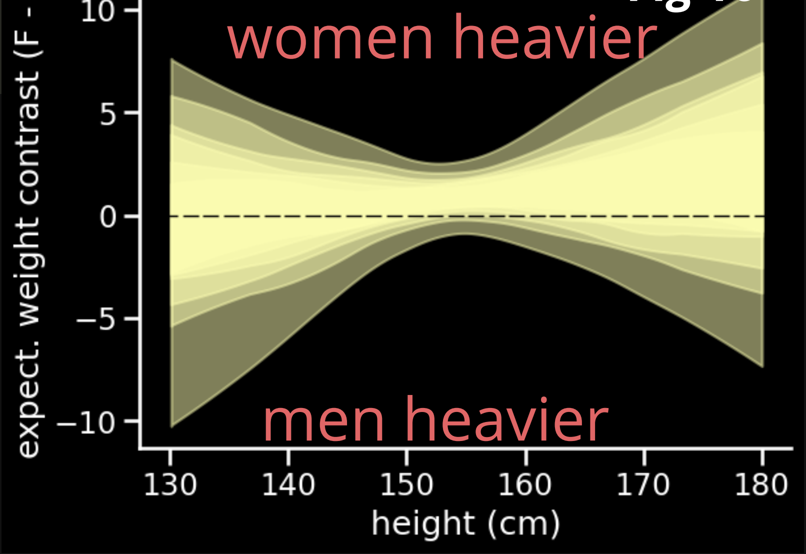


Contrasts at each height

Code 9

```python
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

Fig 10



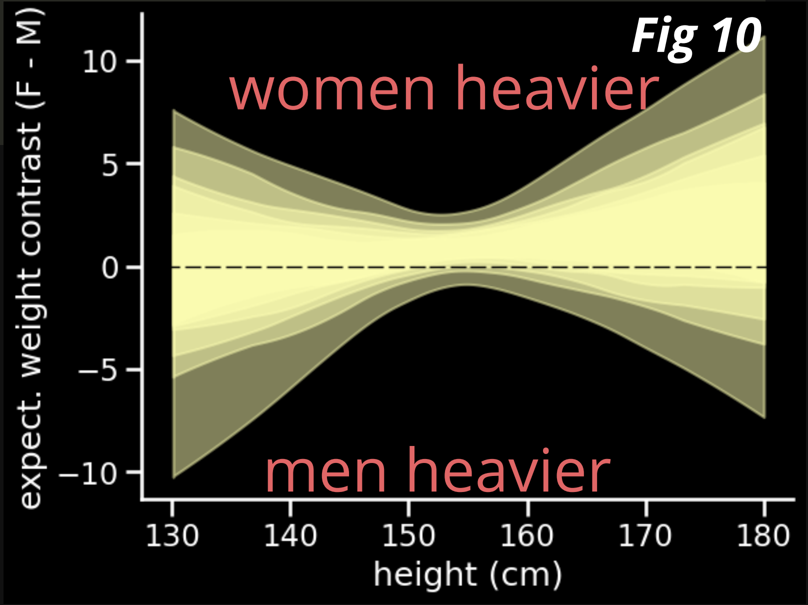
Contrasts at each height

Code 9

```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

contrast distribution *relatively* symmetric around value for no difference (0)

Fig 10

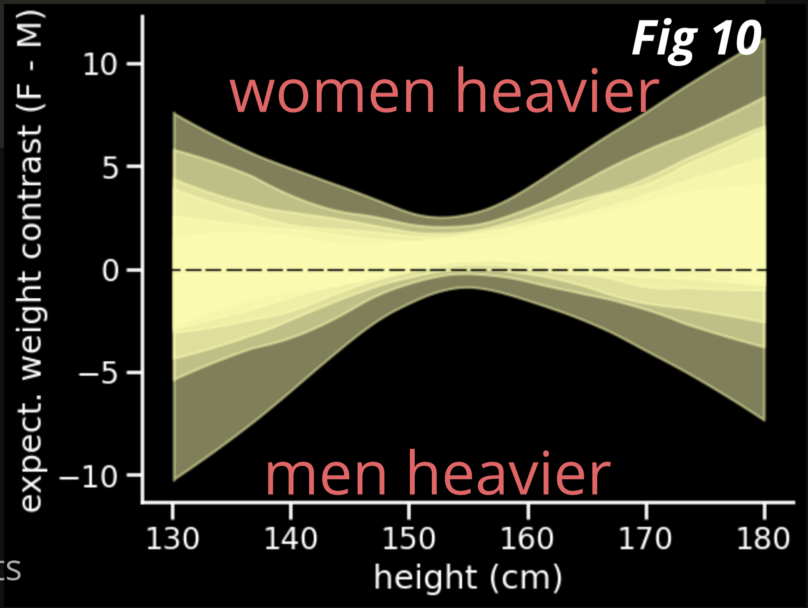


Contrasts at each height

Code 9

```python
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

contrast distribution *relatively* symmetric around value for no difference (0)



less symmetry at extreme heights

Contrasts at each height

Code 9

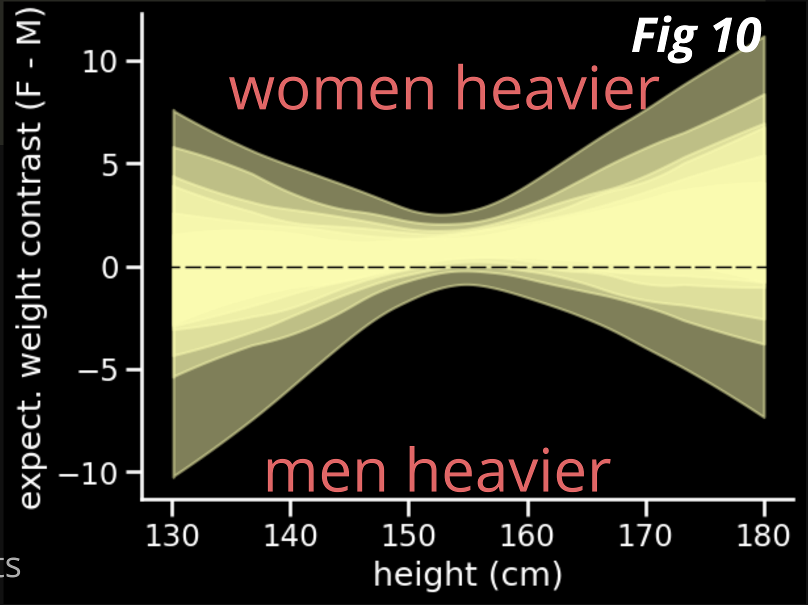
```
import arviz as az

height_seq = np.linspace(130, 180, 30)

data_thin = az.extract(m_GHW_idata, num_samples = 1000)
mu_pred_W = np.empty((len(height_seq), data_thin.sizes["sample"]))
mu_pred_M = np.empty((len(height_seq), data_thin.sizes["sample"]))

for i, ht in enumerate(height_seq):
    mu_pred_W[i] = data_thin.a.values[0] + data_thin.b.values[0] * (ht - df2.height.mean())
    mu_pred_M[i] = data_thin.a.values[1] + data_thin.b.values[1] * (ht - df2.height.mean())

mu_pred_contrast = mu_pred_W - mu_pred_M

for p in [0.5, 0.6, 0.7, 0.8, 0.9, 0.99]:
    _ = az.plot_hdi(height_seq, mu_pred_contrast.T, hdi_prob=p)

_ = plt.plot(height_seq, [0] * len(height_seq), "_", color='k')
_ = plt.xlabel("height (cm)")
_ = plt.ylabel("weight contrast (F - M)")
```

contrast distribution *relatively* symmetric around value for no difference (0)

Nearly all of the causal effect of G acts through H

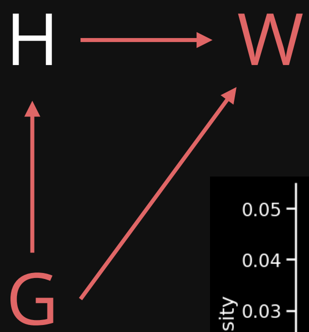
less symmetry at extreme heights

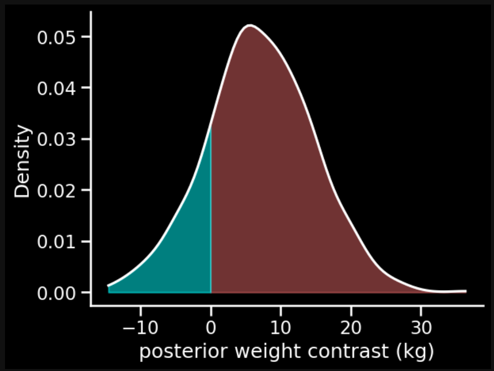
Fig 10

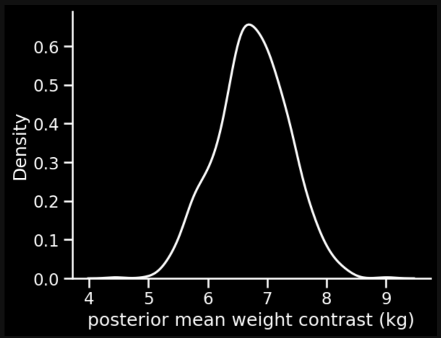


From estimand to estimate

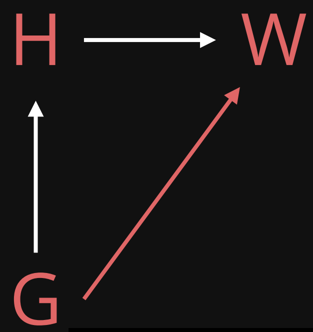
Q: Total causal effect of G on W?

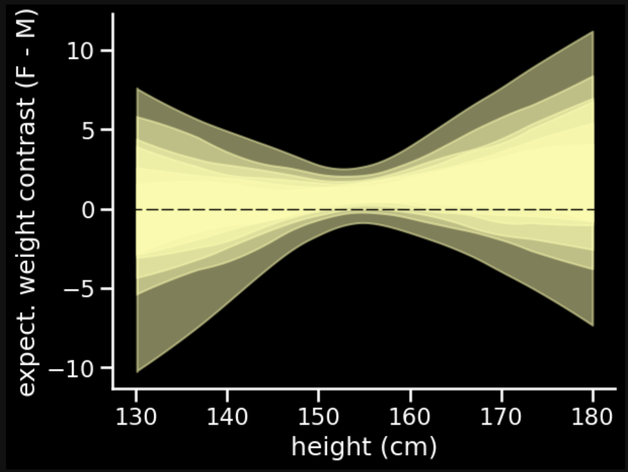






Q: Direct causal effect of G on W?





Summary

Linear regression

(1) Question/goal/estimand

(2) Scientific model

(3) Statistical model

(4) Validate model

(5) Analyze data

Inference with linear models

1. State each estimand

Inference with linear models

With more than two variables, scientific (causal) model and statistical model not always same

1. State each estimand
2. Design **unique statistical model** for each

ONE **STAT MODEL**
FOR EACH ESTIMAND

Inference with linear models

With more than two variables, scientific (causal) model and statistical model not always same

1. State each estimand
2. Design **unique statistical model** for each
3. **Compute** each estimand

ONE **STAT MODEL**
FOR EACH ESTIMAND

Inference with linear models

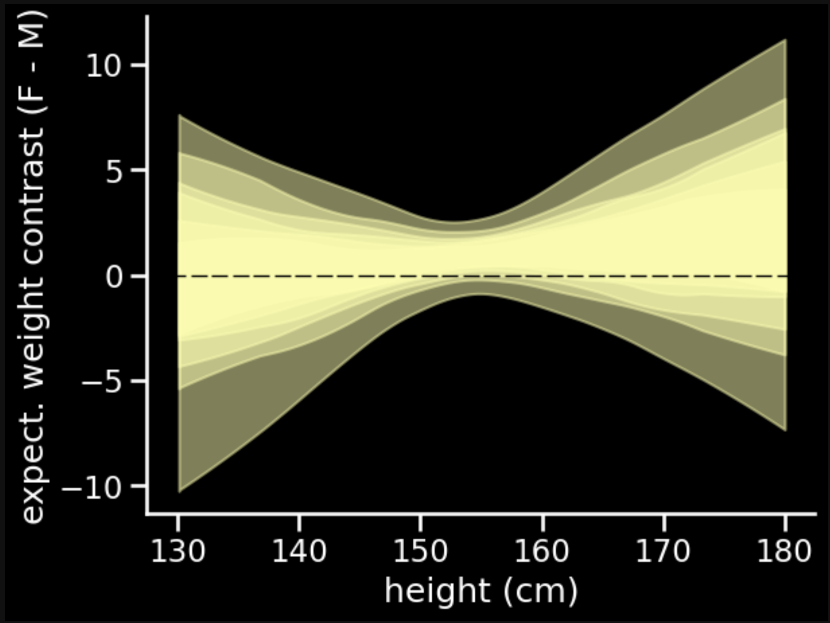
With more than two variables, scientific (causal) model and statistical model not always same

1. State each estimand
2. Design **unique statistical model** for each
3. **Compute** each estimand

ONE **STAT MODEL**
FOR EACH ESTIMAND

Categorical variables

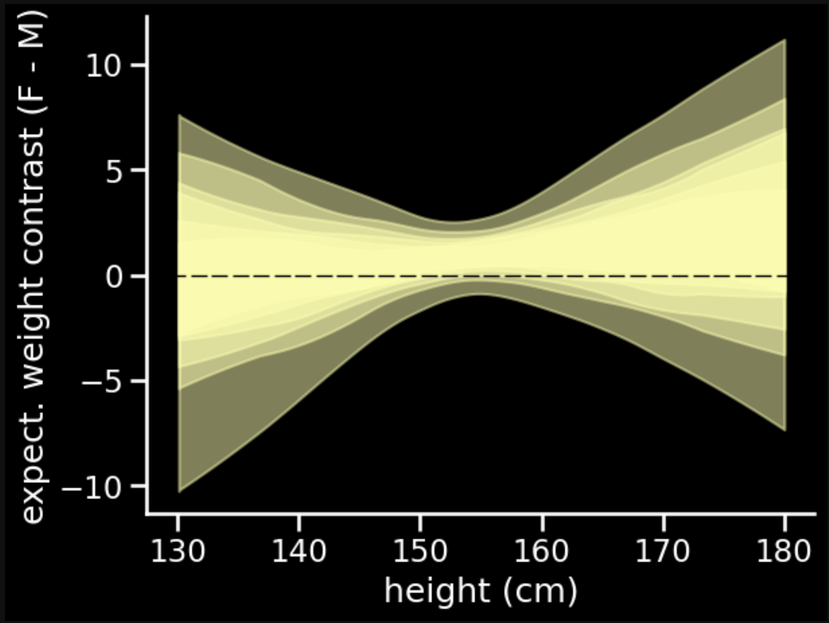
Easy to use with index coding



Categorical variables

Easy to use with index coding

Must later use samples to compute relevant contrasts

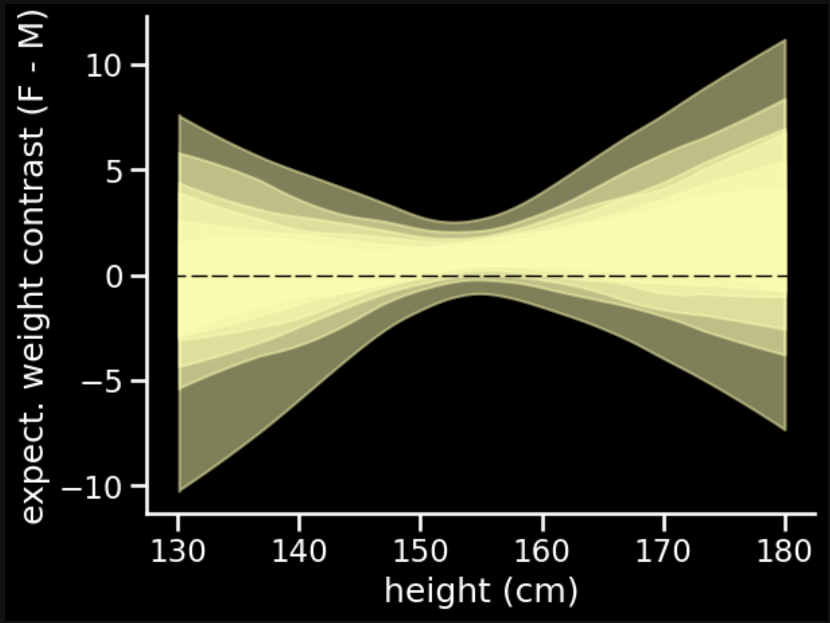


Categorical variables

Easy to use with index coding

Must later use samples to compute relevant contrasts

Always summarize (mean, interval) as last step



Categorical variables

Easy to use with index coding

Must later use samples to compute relevant contrasts

Always summarize (mean, interval) as last step

Want **mean difference** and not **difference of means**