Questions corrigées examens en ligne

Mars 2026

Le jeu de données mpg est inclus dans le package ggplot2. Il contient des informations sur la consommation de carburant de différents modèles de voitures vendus aux États-Unis. Il contient les variables :

- manufacturer : constructeur automobile
- model : nom du modèle
- displ : cylindrée (en litres)
- year : année du modèle
- cyl : nombre de cylindres
- trans : type de transmission
- drv : type de transmission (f = traction avant, r = propulsion arrière, 4 = 4 roues motrices)
- cty : consommation en ville (miles par gallon)
- hwy : consommation sur autoroute (miles par gallon)
- fl : type de carburant
- class : type de véhicule (compact, SUV, etc.)

Le jeu de données iris contient 150 observations de 3 espèces de fleurs : setosa, versicolor, virginic, les variables sont : Species : espèce de la fleur Sepal.Length : longueur du sépale

- Sepal.Width : largeur du sépale
- Petal.Length : longueur du pétale
- Petal.Width : largeur du pétale

examen 2324 - data - 2 pts - 8 minutes

Dans le dataset mpg, trouvez la consommation moyenne sur autoroute pour les voitures à transmission automatique et manuelle, en excluant les voitures dont le type de carburant est diesel.

```
mpg |> filter(fl != "d") |>
group_by(trans) |>
summarise(mean_hwy = mean(hwy, na.rm = TRUE))
```

examen 2324 - data - 2 pts - 8 minutes

Dans le dataset mpg, ajoutez une colonne efficiency calculée comme le rapport de la consommation sur autoroute par la cylindrée. Ensuite, pour chaque manufacturer, trouvez le véhicule le plus efficace (avec la plus grande valeur efficiency). Sélectionnez uniquement les colonnes manufacturer, model, efficiency. Vous pourrez utiliser la fonction slice_max().

```
mpg |> mutate(efficiency = hwy / displ) |>
slice_max(efficiency, n = 1, by = "manufacturer") |>
select(manufacturer, model, efficiency)
```

examen 2425 - data - 6 points - 24 minutes

Dans le dataset mpg, affichez les 5 constructeurs ayant, en moyenne, la meilleure consommation en ville. Pour chacun de ces constructeurs, indiquez le modèle le plus économique en ville. Utilisez dplyr pour votre réponse.

```
top_manufacturers <- mpg |>
group_by(manufacturer) |>
summarise(mean_cty = mean(cty)) |>
arrange(desc(mean_cty)) |>
slice_head(n = 5)

result <- mpg |>
filter(manufacturer %in% top_manufacturers$manufacturer) |>
group_by(manufacturer) |>
slice_max(cty, n = 1) |>
select(manufacturer, model, cty)
```

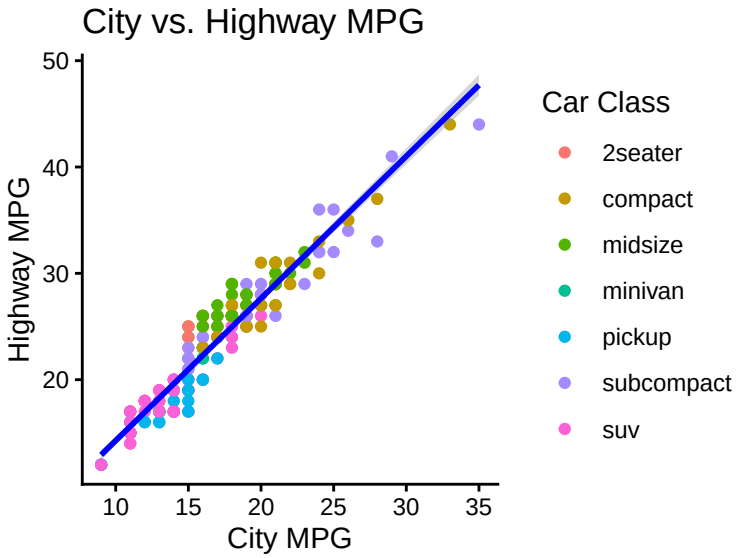
examen 2425 - ggplot - 2 points - 8 minutes

Réalisez un boxplot comparant la longueur des sépales (Sepal.Length) entre les trois espèces. Ajoutez les points individuels sur le même graphique pour mieux visualiser la répartition des données. Personnalisez les axes et le titre du graphique.

```
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot(alpha = 0.6) +
  geom_jitter(width = 0.15, size = 2, alpha = 0.7) +
  labs(
    title = "Comparaison de la longueur des sépales selon l'espèce d'iris",
    x = "Espèce",
    y = "Longueur du sépale") +
  theme_classic()
```

examen 2324 - ggplot - 2 pts - 8 minutes

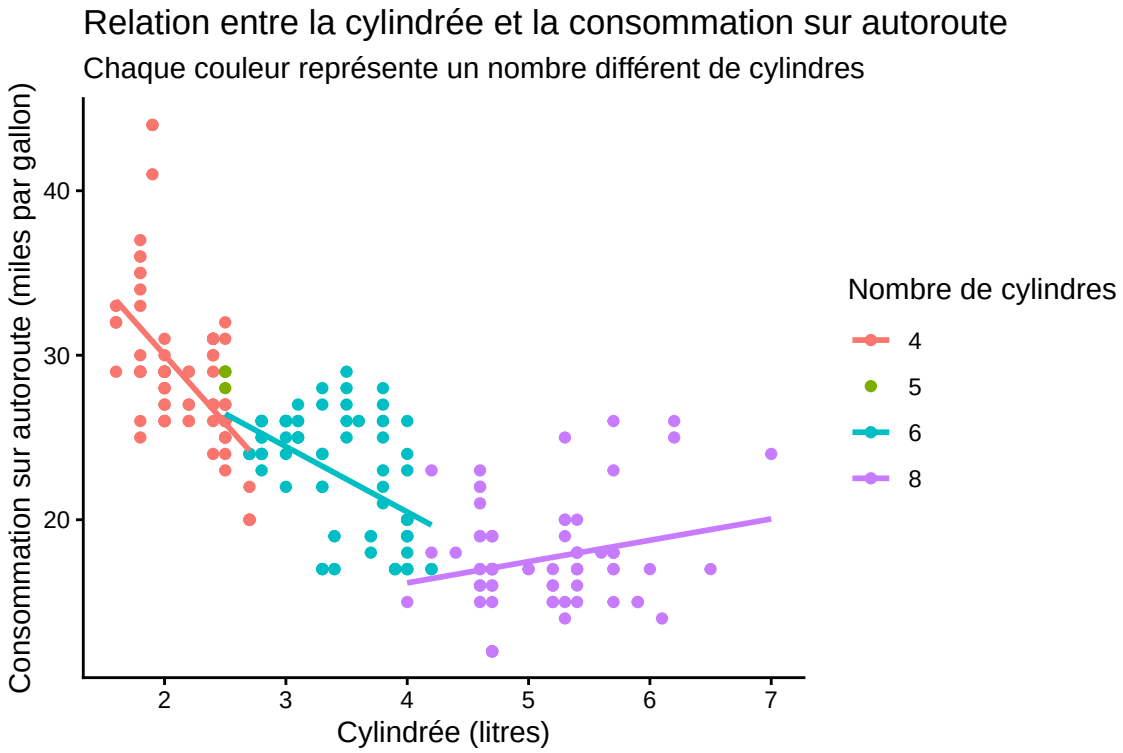
Utilisez le jeu de données mpg pour reproduire le graphique ci-dessous.



```r
ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
  geom_point() +
  geom_smooth(method = "lm", color = "blue") +
  labs(
    title = "City vs. Highway MPG",
    x = "City MPG",
    y = "Highway MPG",
    color = "Car Class") +
  theme_classic()
```

examen 2324 - ggplot - 2 pts - 8 minutes

Utilisez le jeu de données mpg pour reproduire le graphique ci-dessous.




```r
ggplot(mpg, aes(x = displ, y = hwy, color = factor(cyl))) +
  geom_point() + geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Relation entre la cylindrée et la consommation sur autoroute",
    subtitle = "Chaque couleur représente un nombre différent de cylindres",
    x = "Cylindrée (litres)",
    y = "Consommation sur autoroute (miles par gallon)",
    color = "Nombre de cylindres"
  ) +
  theme_classic()
```

examen 2324 - rcpp - 5 pts - 20 minutes

Écrivez une fonction C++ utilisant Rcpp qui prend en entrée un vecteur numérique R et deux seuils. La fonction doit modifier chaque élément du vecteur de la manière suivante : si un élément est inférieur au premier seuil, il doit être réduit de moitié ; si un élément est supérieur au second seuil, il doit être doublé. La fonction retourne le vecteur modifié.

```
cppFunction('
NumericVector modify_vector(NumericVector vec, double threshold1, double threshold2) {
  int n = vec.size();
  for (int i = 0; i < n; ++i) {
    if (vec[i] < threshold1) {
      vec[i] = vec[i] / 2.0;
    } else if (vec[i] > threshold2) {
      vec[i] = vec[i] * 2.0;
    }
  }
  return vec;
}
')

modify_vector(c(12,4,5,10,17,20),6,15)
```

examen 2324 - rcpp - 5 pts - 20 minutes

Écrivez une fonction C++ utilisant Rcpp pour trier un vecteur numérique R en utilisant l'algorithme de tri par sélection. Cette méthode consiste à trouver le plus petit élément du vecteur et à le placer au début, puis à répéter cette opération pour la sous-liste restante jusqu'à ce que le vecteur soit entièrement trié. La fonction retourne le vecteur trié.

```
cppFunction('
NumericVector sort(NumericVector vec) {
  int n = vec.size();
  for (int i = 0; i < n - 1; ++i) {
    int min_idx = i;
    for (int j = i + 1; j < n; ++j) {
      if (vec[j] < vec[min_idx]) {
        min_idx = j;
      }
    }
    if (min_idx != i) {
      double temp = vec[i];
      vec[i] = vec[min_idx];
      vec[min_idx] = temp;
    }
  }
  return vec;
}
')

sort(c(29,10,14,37,15))
```

examen 2425 - rcpp - 5 points - 20 minutes

Écrivez une fonction C++ utilisant Rcpp qui prend en entrée un vecteur numérique et retourne un nouveau vecteur où chaque élément est remplacé par la moyenne de lui-même et de ses deux voisins (gauche et droite). Les bords (premier et dernier élément) restent inchangés.

```
cppFunction('
NumericVector moyenne_voisins(NumericVector x) {
  int n = x.size();
  if (n <= 2) return clone(x);
  NumericVector res(n);
  res[0] = x[0];
  res[n - 1] = x[n - 1];
  for (int i = 1; i < n - 1; i++) res[i] = (x[i - 1] + x[i] + x[i + 1]) / 3.0;
  return res;
}
')

moyenne_voisins(c(13,2,3,4,56,6))
```