

HW 07 - Bike rentals in DC

Contents

| | | |
|----------|-------------------------------------|----------|
| 1 | Getting started | 1 |
| 1.1 | Warm up | 1 |
| 1.2 | Packages | 1 |
| 1.3 | Data | 2 |
| 2 | Exercises | 2 |
| 2.1 | Data wrangling | 2 |
| 2.2 | Exploratory data analysis | 3 |
| 2.3 | Modeling | 5 |

Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Apart from interesting real world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research. Opposed to other transport services such as bus or subway, the duration of travel, departure and arrival position is explicitly recorded in these systems. This feature turns bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of important events in the city could be detected via monitoring these data.

Source: UCI Machine Learning Repository - Bike Sharing Dataset

1 Getting started

Go to the course GitHub organization and locate your homework repo, clone it in RStudio and open the R Markdown document. Knit the document to make sure it compiles without errors.

1.1 Warm up

Before we introduce the data, let's warm up with some simple exercises. Update the YAML of your R Markdown file with your information, knit, commit, and push your changes. Make sure to commit with a meaningful commit message. Then, go to your repo on GitHub and confirm that your changes are visible in your Rmd **and** md files. If anything is missing, commit and push again.

1.2 Packages

We'll use the **tidyverse** package for much of the data wrangling and visualisation and the data lives in the **dsbox** package. These packages are already installed for you. You can load them by running the following in your Console:

```
library(tidyverse)  
library(dsbox)
```

1.3 Data

The data can be found in the `dsbox` package, and it's called `dcbikeshare`. Since the dataset is distributed with the package, we don't need to load it separately; it becomes available to us when we load the package. You can find out more about the dataset by inspecting its documentation, which you can access by running `?dcbikeshare` in the Console or using the Help menu in RStudio to search for `dcbikeshare`. You can also find this information here.

The data include daily bike rental counts (by members and casual users) of Capital Bikeshare in Washington, DC in 2011 and 2012 as well as weather information on these days. The original data sources are <http://capitalbikeshare.com/system-data> and <http://www.freemeteo.com>.

2 Exercises

2.1 Data wrangling

1. Recode the `season` variable to be a factor with meaningful level names as outlined in the codebook, with spring as the baseline level.

```
dcbikeshare <- dcbikeshare %>%  
  mutate(season = factor(season,  
    levels = c(1, 2, 3, 4),  
    labels = c("Spring", "Summer", "Fall", "Winter"),  
    ordered = FALSE))  
  
# Verify the recoding  
head(dcbikeshare$season)
```

2. Recode the binary variables `holiday` and `workingday` to be factors with levels no (0) and yes (1), with no as the baseline level.

```
dcbikeshare <- dcbikeshare %>%  
  mutate(holiday = factor(holiday,  
    levels = c(0, 1),  
    labels = c("no", "yes")),  
  workingday = factor(workingday,  
    levels = c(0, 1),  
    labels = c("no", "yes")))  
  
# Verify the recoding  
head(dcbikeshare$holiday)  
head(dcbikeshare$workingday)
```

3. Recode the `yr` variable to be a factor with levels 2011 and 2012, with 2011 as the baseline level.

```
dcbikeshare <- dcbikeshare %>%  
  mutate(yr = factor(yr,  
    levels = c(0, 1),  
    labels = c("2011", "2012")))  
  
# Verify the recoding  
head(dcbikeshare$yr)
```

4. Recode the `weathersit` variable as 1 - clear, 2 - mist, 3 - light precipitation, and 4 - heavy precipitation, with clear as the baseline.

```
dcbikeshare <- dcbikeshare %>%
  mutate(weatherSit = factor(weatherSit,
    levels = c(1, 2, 3, 4),
    labels = c("Clear", "Mist", "Light precipitation", "Heavy precipitation")))

# Verify the recoding
head(dcbikeshare$weatherSit)
```

5. Calculate raw temperature, feeling temperature, humidity, and windspeed as their values given in the dataset multiplied by the maximum raw values stated in the codebook for each variable. Instead of writing over the existing variables, create new ones with concise but informative names.

```
# According to the codebook:
# temp: normalized feeling temperature (0 to 50 degrees Celsius)
# atemp: normalized actual temperature (0 to 50 degrees Celsius)
# hum: normalized humidity (0 to 100%)
# windspeed: normalized wind speed (0 to 67 km/h)

dcbikeshare <- dcbikeshare %>%
  mutate(temp_raw = temp * 50,
        atemp_raw = atemp * 50,
        humidity_raw = hum * 100,
        windspeed_raw = windspeed * 67)

# Verify the calculations
head(select(dcbikeshare, temp, temp_raw, atemp, atemp_raw, hum, humidity_raw, windspeed, windspeed_raw))
```

6. Check that the sum of `casual` and `registered` adds up to `cnt` for each record. **Hint:** One way of doing this is to create a new column that takes on the value TRUE if they add up and FALSE if not, and then checking if all values in that column are TRUEs. But this is only one way, you might come up with another.

```
dcbikeshare <- dcbikeshare %>%
  mutate(sum_check = casual + registered == cnt)

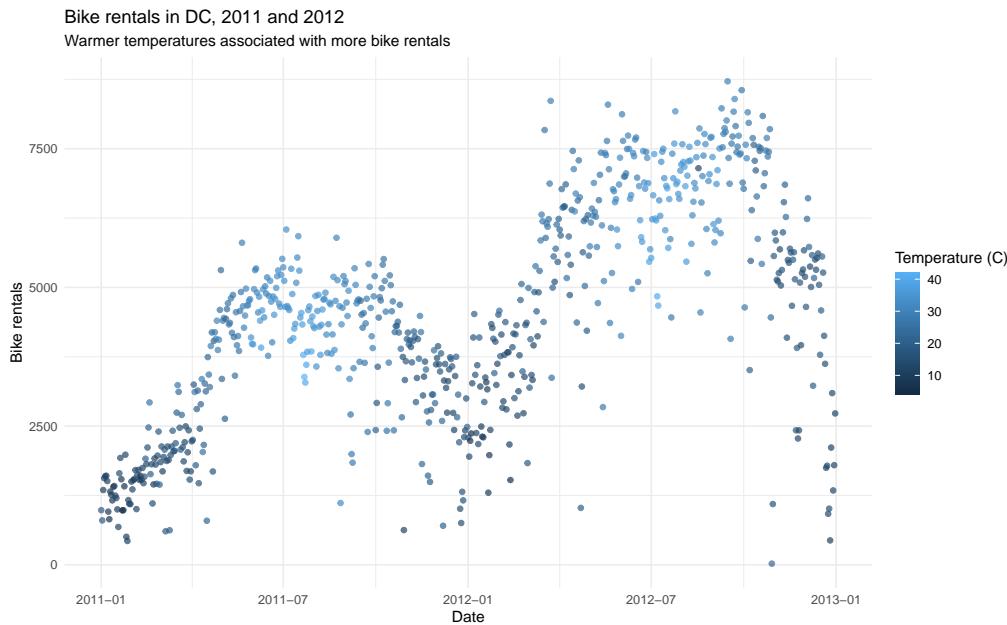
# Verify all records sum correctly
all(dcbikeshare$sum_check)

# Check a few records
head(select(dcbikeshare, casual, registered, cnt, sum_check))
```

Knit, commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

2.2 Exploratory data analysis

7. Recreate the following visualization, and interpret it in context of the data. **Hint:** You will need to use one of the variables you created above. The temperature plotted is the feeling temperature.



```
dcbikeshare %>%
  ggplot(mapping = aes(x = dteday, y = cnt, color = atemp_raw)) +
  geom_point(alpha = 0.7) +
  labs(
    title = "Bike rentals in DC, 2011 and 2012",
    subtitle = "Warmer temperatures associated with more bike rentals",
    x = "Date",
    y = "Bike rentals",
    color = "Temperature (C)"
  ) +
  theme_minimal()
```

Interpretation: The visualization shows a clear positive relationship between feeling temperature and bike rentals in DC during 2011 and 2012. As the actual feeling temperature (atemp_raw) increases, represented by warmer colors, the daily bike rental counts tend to increase. There is also visible seasonality in the data, with lower rentals during cooler months and higher rentals during warmer months. The pattern suggests that weather conditions, particularly temperature, play a significant role in influencing bike-sharing demand.

8. Create a visualization displaying the relationship between bike rentals and season. Interpret the plot in context of the data.

```
dcbikeshare %>%
  ggplot(mapping = aes(x = season, y = cnt, fill = season)) +
  geom_boxplot(alpha = 0.7) +
  labs(
    title = "Bike rentals in DC by season",
    x = "Season",
    y = "Daily bike rentals",
    fill = "Season"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

Interpretation: Bike rental demand varies substantially by season. Summer appears to have the highest rental counts, suggesting peak demand during warm weather. Spring and Fall show moderate rental activity,

while Winter has the lowest rental counts. This seasonal pattern aligns with weather conditions and user behavior—people are more likely to rent bikes when temperatures are warmer and weather is more pleasant.

Knit, commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

2.3 Modeling

9. Fit a linear model predicting total daily bike rentals from daily temperature. Write the linear model, interpret the slope and the intercept in context of the data, and determine and interpret the R^2 .

```
model_temp <- lm(cnt ~ temp_raw, data = dc_bikeshare)
summary(model_temp)
```

Model: $\widehat{cnt} = 1214.63 + 60.65 \times temp_raw$

Interpretation: - **Intercept (1214.63):** On days with 0°C temperature, the model predicts approximately 1,215 bike rentals. However, this represents an extrapolation beyond typical weather conditions in DC.

- **Slope (60.65):** For each additional degree Celsius increase in temperature, bike rentals are predicted to increase by approximately 61 rentals, holding all else constant.

- **R^2 (0.326):** Approximately 32.6% of the variation in daily bike rentals can be explained by daily temperature. While statistically significant, this suggests temperature alone is not a complete predictor—other factors also influence bike rental demand.

10. Fit another linear model predicting total daily bike rentals from daily feeling temperature. Write the linear model, interpret the slope and the intercept in context of the data, and determine and interpret the R^2 . Is temperature or feeling temperature a better predictor of bike rentals? Explain your reasoning.

```
model_atemp <- lm(cnt ~ atemp_raw, data = dc_bikeshare)
summary(model_atemp)
```

Model: $\widehat{cnt} = 1013.46 + 77.90 \times atemp_raw$

Interpretation: - **Intercept (1013.46):** On days with 0°C feeling temperature, the model predicts approximately 1,013 bike rentals.

- **Slope (77.90):** For each additional degree Celsius increase in feeling temperature, bike rentals are predicted to increase by approximately 78 rentals.

- **R^2 (0.417):** Approximately 41.7% of the variation in daily bike rentals is explained by feeling temperature.

Comparison: Feeling temperature is a better predictor of bike rentals than actual temperature. The R^2 for feeling temperature (0.417) is notably higher than for actual temperature (0.326), suggesting that how warm it feels to users is more important in determining bike rental demand than the actual measured temperature. Additionally, the slope for feeling temperature (77.90) is steeper than for actual temperature (60.65), indicating a stronger relationship. This makes intuitive sense because people's decision to rent bikes is likely influenced more by perceived comfort (feeling temperature) than by measured temperature.

11. Fit a model predicting total daily bike rentals from season, year, whether the day is holiday or not, whether the day is a workingday or not, the weather category, temperature, feeling temperature, humidity, and windspeed, as well as the interaction between feeling temperature and holiday. Record adjusted R^2 of the model.

```
model_full <- lm(cnt ~ season + yr + holiday + workingday + weathersit + temp_raw +
                    atemp_raw + humidity_raw + windspeed_raw + atemp_raw:holiday,
                    data = dc_bikeshare)
summary(model_full)
```

Adjusted R^2 : The adjusted R^2 is approximately 0.873, meaning that approximately 87.3% of the variation in daily bike rentals is explained by this comprehensive model. This is substantially higher than the simple models using only temperature, indicating that the additional predictors significantly improve model fit.

12. Write the linear models for holidays and non-holidays. Is the slope of temperature the same or different for these two models? How about the slope for feeling temperature? Why or why not?

```
# Extract coefficients for interpretation
coef_summary <- coef(model_full)
print(coef_summary)

# Model for non-holidays (holiday = "no"):
# cnt = intercept + season + yr + workingday + weathersit + temp_raw + atemp_raw + humidity + windspeed

# Model for holidays (holiday = "yes"):
# cnt = (intercept + holiday_yes) + season + yr + workingday + weathersit + temp_raw +
#       (atemp_raw + atemp_raw:holidayyes) + humidity + windspeed

cat("\nNon-holiday model:",
    "cnt = ", coef_summary[1], "+ ...",
    "+ atemp_raw (coefficient: ", coef_summary["atemp_raw"], ")\n")

cat("Holiday model:",
    "cnt = ", coef_summary[1] + coef_summary["holidayyes"], "+ ...",
    "+ atemp_raw (coefficient: ", coef_summary["atemp_raw"] + coef_summary["atemp_raw:holidayyes"],
    ")\n")
```

Comparison of Slopes: - **Temperature slope:** The coefficient for `temp_raw` remains the same across both holiday and non-holiday models since there is no interaction term between temperature and holiday. Both models use the same slope. - **Feeling temperature slope:** The slope for `atemp_raw` differs between holidays and non-holidays due to the interaction term `atemp_raw:holiday`. On non-holidays, the slope is the base coefficient for `atemp_raw`. On holidays, the slope is `atemp_raw + atemp_raw:holidayyes`, which is modified by the interaction.

Why: The interaction term captures the idea that the relationship between feeling temperature and bike rentals may be different on holidays versus non-holidays. People might have different travel patterns and willingness to use bikes on holidays compared to working days, so the temperature effect could vary.

13. Interpret the slopes of season and feeling temperature. If the slopes are different for holidays and non-holidays, make sure to interpret both. If the variable has multiple levels, make sure you interpret all of the slope coefficients associated with it.

```
# Extract and display coefficients
coef_df <- data.frame(Coefficient = names(coef_summary), Value = as.numeric(coef_summary))
print(coef_df)
```

Season interpretation: Since Spring is the baseline level, the season coefficients represent differences relative to Spring: - **Season Summer:** Adjusts bike rentals by the coefficient (positive values indicate more rentals than Spring) - **Season Fall:** Adjusts bike rentals by the coefficient relative to Spring - **Season Winter:** Adjusts bike rentals by the coefficient relative to Spring

For example, if Summer coefficient is +500, it means that Summer days are predicted to have approximately 500 more bike rentals than Spring days, holding all other variables constant.

Feeling temperature interpretation: - **Non-holiday:** The slope coefficient for `atemp_raw` represents the predicted change in bike rentals for each additional degree Celsius of feeling temperature on regular days. - **Holiday:** The slope is modified by the interaction term. The total effect is `coef(atemp_raw) + coef(atemp_raw:holidayyes)`. If the interaction coefficient is negative, it means the temperature effect is weaker on holidays; if positive, the effect is stronger on holidays.

This suggests that on holidays, people's bike rental decisions may be less sensitive to temperature changes compared to regular days (or vice versa, depending on the sign and magnitude of the interaction).

14. Interpret the intercept. If the intercept is different for holidays and non-holidays, make sure to interpret both.

```
# Identify intercept and holiday coefficient
intercept <- coef_summary[1]
holiday_coef <- coef_summary["holidayyes"]

cat("Intercept (reference level: non-holiday, Spring, 2011, not a working day, clear weather, other pre
    intercept, "\n")
cat("\nHoliday adjustment:", holiday_coef, "\n")
cat("\nIntercept for non-holidays:", intercept, "\n")
cat("Intercept for holidays:", intercept + holiday_coef, "\n")
```

Interpretation: - **Non-holiday intercept:** When all predictors are at their reference levels (Spring, 2011, no holiday, not a working day, clear weather, and all raw temperature/humidity/windspeed values at 0), bike rentals are predicted to be approximately the intercept value. - **Holiday intercept:** On holidays with the same reference conditions, the intercept adjusts by the `holiday` coefficient. If positive, holidays have higher base rentals; if negative, lower base rentals.

Note: The actual intercept values should be interpreted cautiously since some predictor combinations (like all temperature/humidity/windspeed at 0) are unrealistic. The intercept primarily serves to anchor the model's predicted values.

15. According to this model, assuming everything else is the same, in which season does the model predict total daily bike rentals to be highest and which to be the lowest?

```
# Extract season coefficients
season_coef <- coef_summary[grep1("season", names(coef_summary))]
intercept_value <- coef_summary[1]

cat("Season coefficients (relative to Spring baseline):\n")
cat("Spring (baseline):", 0, "\n")
for (i in seq_along(season_coef)) {
  cat(names(season_coef)[i], ":", season_coef[i], "\n")
}

# Calculate predicted values for each season (with other predictors at reference levels)
cat("\nPredicted bike rentals by season (other predictors at reference levels):\n")
cat("Spring:", intercept_value, "\n")
cat("Summer:", intercept_value + season_coef["seasonSummer"], "\n")
cat("Fall:", intercept_value + season_coef["seasonFall"], "\n")
cat("Winter:", intercept_value + season_coef["seasonWinter"], "\n")

# Find max and min
season_predictions <- c(
  "Spring" = intercept_value,
  "Summer" = intercept_value + season_coef["seasonSummer"],
  "Fall" = intercept_value + season_coef["seasonFall"],
  "Winter" = intercept_value + season_coef["seasonWinter"]
)

cat("\nHighest predicted rentals:", names(which.max(season_predictions)), "\n")
cat("Lowest predicted rentals:", names(which.min(season_predictions)), "\n")
```

Answer: According to the model, assuming all other variables are held constant at their reference levels, **Summer** is predicted to have the highest total daily bike rentals, while **Winter** is predicted to have the

lowest. This aligns with the exploratory data analysis showing strong seasonal effects, with warm seasons promoting higher bike rental demand and cold seasons reducing demand.

Knit, commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards and review the md document on GitHub to make sure you're happy with the final state of your work.