Tasks chapter5:

David Kurz, Malte Hildebrandt
2023S707737 VU Geostatistik
Universität Innsbruck

April 25, 2023

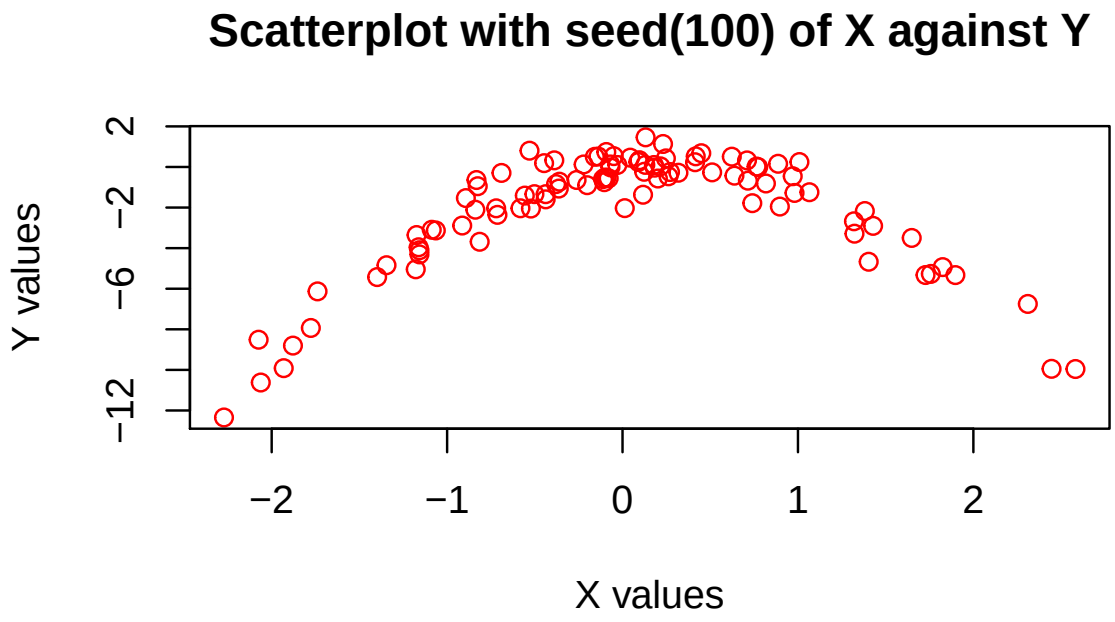
1 resampling: Cross-validation

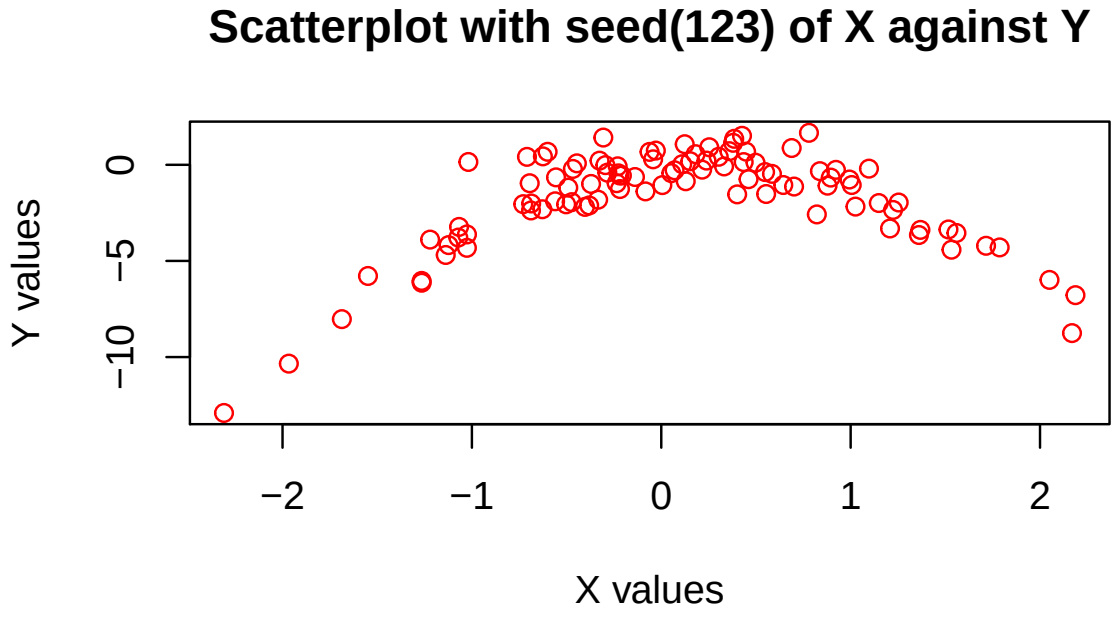
1. **Generate a simulated data set:**
   set.seed(100)
   x <- rnorm(100)
   y <- x - 2*x^2 + rnorm(100)
   What is n and what is p in this data set? Write the last line (y <- ...) as a mathematical equation for linear regression. What does rnorm(100) represent in this equation? Remember that you can find out what the functions rnorm and set.seed do by typing ?rnorm in the R console (or googling).

   - n is number of observations. If length(n) > 1, the length is taken to be the number required.
   - p is the vector of probabilities.
   - $rnorm(100)$: rnorm generates random deviates with 100 observations.
   - ?$rnrom$ in R-console:
     Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.
   - ?$set.seed$ in R-console: is a Random Number Generation where random.seed is an integer vector, containing the random number generator (RNG) state for random number generation in R. It can be saved and restored, but should not be altered by the user.

---

2. **Data scatterplot:**
   Produce a scatterplot of X against Y to get a visual impression of the data you created.


Scatterplot with seed(100) of X against Y
Y values
2
−2
−6
−12
−2
−1
0
1
2
X values


Scatterplot with seed(123) of X against Y
Y values
0
−5
−10
−2
−1
0
1
2
X values

3. **LLS regression:**
Set a random seed and compute LOOCV errors that result from fitting four models using least squares:

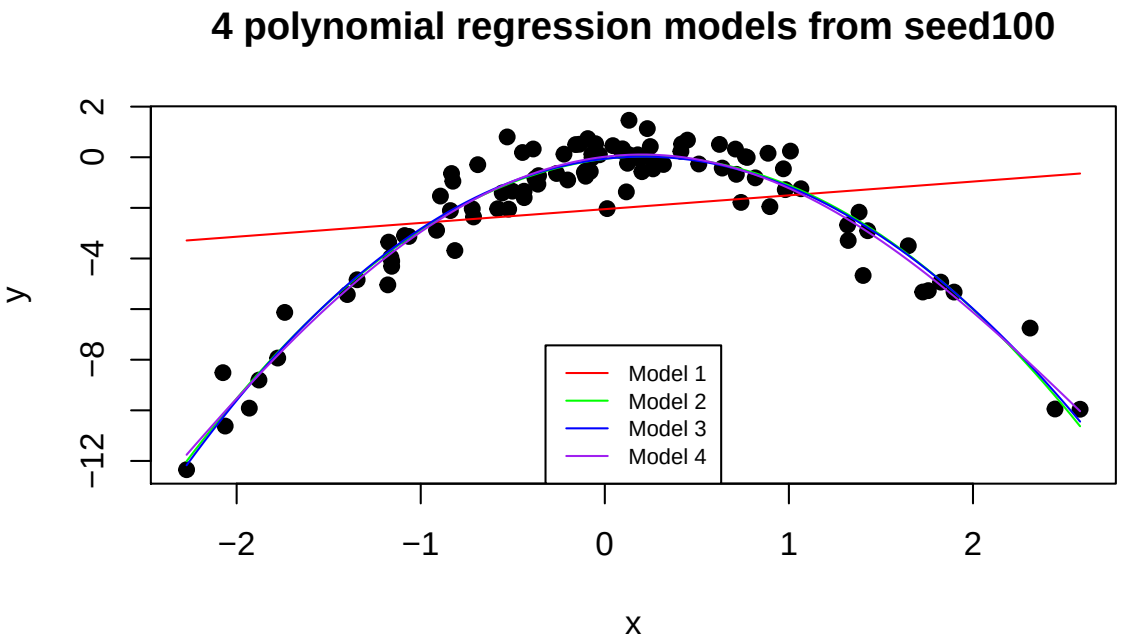
Y = b0 + b1*X + epsilon
Y = b0 + b1*X + b2*x^2 + epsilon
Y = b0 + b1*X + b2*x^2 + b3*x^3 + epsilon
Y = b0 + b1*X + b2*x^2 + b3*x^3 + b4*x^4 + epsilon

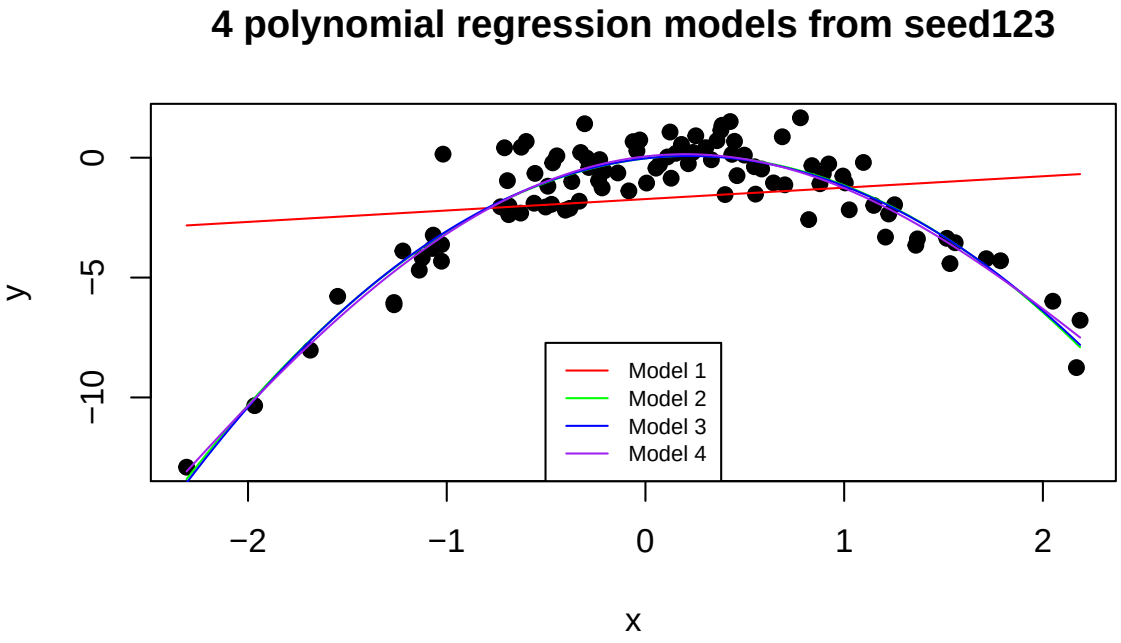
Use data.frame() to create one data set that contains X and Y. Use glm() with the option family = "gaussian" to fit the linear model instead of lm(). The reason is that its results can be used with cv.glm() to compute cross validation. cv.glm is contained in the boot package (which you must install if not already on your system and then load). For LOOCV you only need to specify your data set and the variable to which you assigned your glm-fitted model as parameters for the cv.glm function (cf. also section 5.3.2 in James2021). Careful! Formulas in R do not evaluate their contents. For example, in y ∼ x + x^2, R would interpret the second term as a duplicate of x and drop it. You need to use the "as-is operator" I(): y ∼ x + I(x^2), which tells R to compute the values of x^2 before attempting to use the formula.

```
> print(paste0("LOOCV error for model 1: ", cv_error1))
[1] "LOOCV error for model 1: 9.060636163442"
> print(paste0("LOOCV error for model 2: ", cv_error2))
[1] "LOOCV error for model 2: 0.651190888467549"
> print(paste0("LOOCV error for model 3: ", cv_error3))
[1] "LOOCV error for model 3: 0.666533937312975"
> print(paste0("LOOCV error for model 4: ", cv_error4))
[1] "LOOCV error for model 4: 0.667126097349285"
```

```
> print(paste0("LOOCV error for model 1: ", cv_error1))
[1] "LOOCV error for model 1: 6.97521182500645"
> print(paste0("LOOCV error for model 2: ", cv_error2))
[1] "LOOCV error for model 2: 0.966467776168288"
> print(paste0("LOOCV error for model 3: ", cv_error3))
[1] "LOOCV error for model 3: 1.00001743063605"
> print(paste0("LOOCV error for model 4: ", cv_error4))
[1] "LOOCV error for model 4: 0.99932153669579"
```

4. **Interpret LOOCV errors:**
Which of the 4 models has the smallest LOOCV error? Plot all 4 models (resulting from the seed in subtask 3) on top of the 100 data points to explain the results





- Model 2 with seed(100) has the lowest LOOC error with rounded value of 0.65.
- Model 2 with seed(123) has the lowest LOOC error with rounded value of 0.97.

---

5. **Regenerate data set and regressions:**
   Repeat subtasks 1 - 3 but with a different random seed. Do the results differ? Why/not?

   See Plots...
   Results:

   - Model 2 has the lowest LOOCV error with both seeds
   - Model 3 and 4 are similar to Model 2
   - Model 1 has the highest LOOCV errors (linear fit)

---

2 Bootstrap: classification of wet and drenched days and bootstrapped ROC of its performance

1. **Fit logistic regression models:**
   Analogously to the task of chapter 4, fit a logistic regression model to mornings with measurable precipitation (location$wet) using even mornings (i.e. mornings of days 2, 4, 6, 8, . . .) for training and assign it to trainLocation, and use odd mornings (of days 1, 3, 5, 7, . . .) for testing (assign to locationTest). You can find even and odd days with the modulo function %% 2. Replace "location" with the location assigned to you.

   Repeat for mornings with at least 12 mm of precipitation (ibk$drenched). Remark: since you have to perform 2 very similar tasks that differ only in the precipitation threshold, you might want to write function(s) to achieve them. This is optional but would be good programming practice. For a good and simple introduction on how to write functions in R, see https://discdown.org/rprogramming/functions.html.

   Just programming in R

---

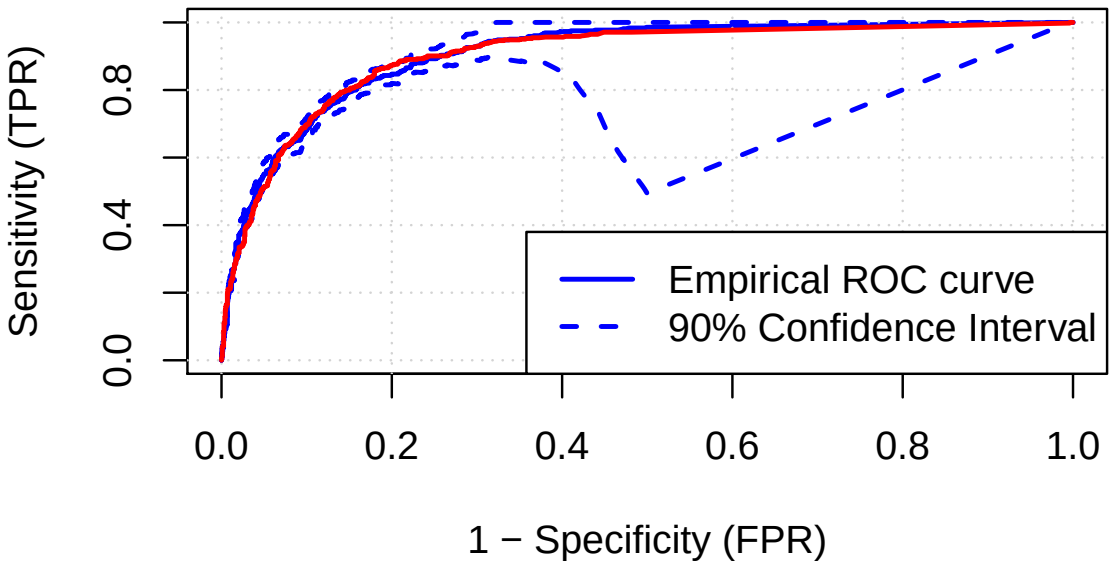
2. **Plot the ROC curve for the training data set:**
   with 90% confidence interval bands obtained by bootstrap, one figure for each of wet and drenched mornings. Use the package ROCit to compute the ROC curve. Comment on the width of the confidence interval. The vignette of the package gives a good overview of how to use it: https://cran.r-project.org/web/packages/ROCit/vignettes/my-vignette.html.
   First fit a roc-object with rocEmp <- rocit() using the empirical method. class are the data you used for classifying the data, i.e. the observed wet (or drenched) vs. dry mornings; score is your predicted model (here: predictions on the training data).

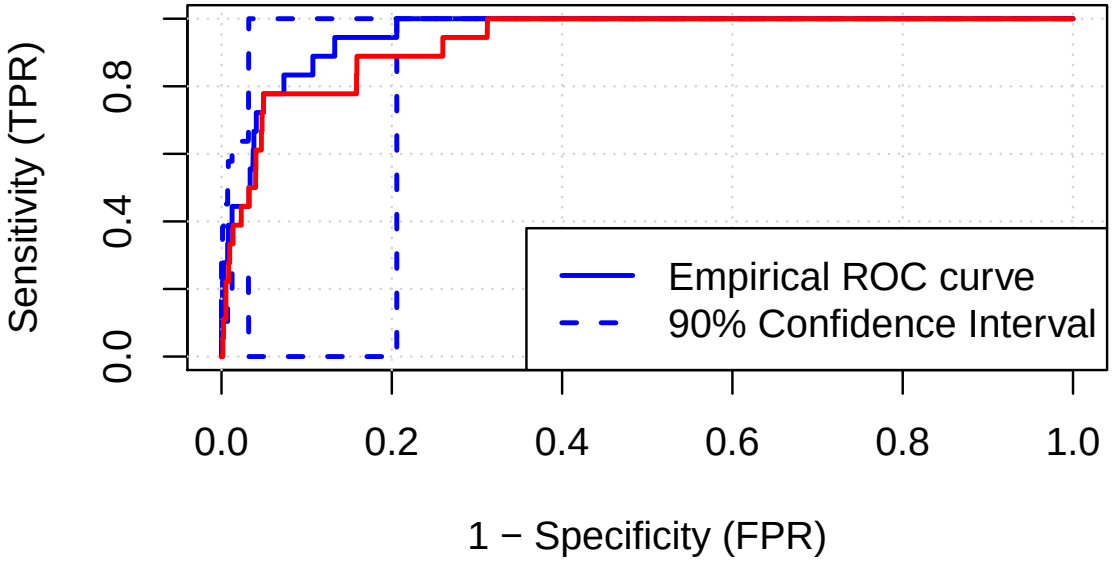
Then compute the 90% confidence interval with roc90 <- ciROC() using 100 bootstrap samples. Finally, use plot(roc90) with any line/color options of your choice. Note that not all functions of ROCit can handle zoo-objects (the format our data set is in). If you get an error message you can try again by stripping the date/time information from the offending variable using coredata(offendingVariable). You may, however, ignore warning messages.

Just programming in R

---

3. **Add the test data ROC curve:**
Add the ROC curve without confidence intervals for the prediction for the test data period to the previous plot. Why does it (not) fall within the 90% confidence intervals?



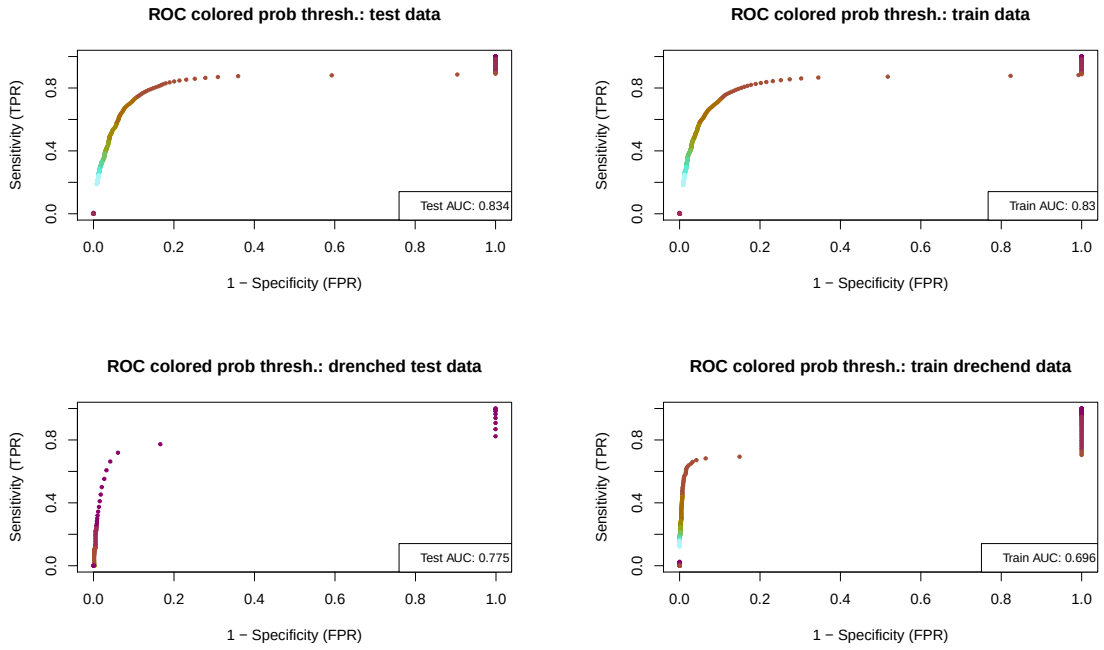


The red curve is the ROC-curve of the test data set. For wet days it lies inbetween the 90% confidence interval from the trained data set. If we consider days with more or equal to 12 mm precipitation, the red ROC-curve falls out of the 90% confidence interval.
The confidence intervals are based on the sample size, the variability of the data, and the level of confidence chosen.If the red ROC curve of the test data falls within the confidence intervals of the trained data, it suggests that the model's performance is consistent with what we would expect given the variability of the data. If the red ROC curve falls outside the confidence intervals of trained data, it suggests that the model's performance is either better or worse than expected given the variability of the data. In our case the red curve of test data suggest a worse model performance. The model performs poorly for the drained dataset because these days do not occur cyclically and therefore a linear model is less able to describe them. Therefore some of those values fall out of the 90% confidence interval.

---

4. **ROC with probability thresholds:**
The ROCit package also provides the probability thresholds used to plot the ROC curve. Its name in the package is "Cutoff". Exploit that to plot the ROC curve color-coded by the probability threshold values. Use a color palette from the colorspace package (https://cran.r-project.org/web/packages/colorspace/vignettes/colorspace.html). Use a sequential multihue palette with 10 color bands: colRoc <- sequential_hcl(10, palette = "Hawai"). Use rocNp <- rocit(..., method = "non") to create a roc-object fit with the non-parametric method. Add a color variable to the rocNp object: rocNp$col. The color is determined by the threshold probability

contained in rocNp$Cutoff. For example, elements of rocNp with probability thresholds between 0 and 0.1 would be assigned the first color band of colRoc; thresholds between 0.9 and 1 to the last color band. Note, though, that the ROCit package also uses probability threshold values outside of [0, 1], which you will have to treat appropriately. Use plot(..., type = p, pch = 19, cex = 0.5) to plot the color-coded ROC-curve (and investigate what type and pch options do).



For the normal wet mornings, the AUC values are quite similar and, at 0.8, not very high by meteorological standards. This is because the prediction depends on the previous morning (linear through the model).
For the drenched days, the trained dataset performs worse with a lower AUC (0.775 test vs 0.696 trained). This is due to the fact that a prediction based on the previous morning is imprecise for such precipitation totals. For this, other predictors and time points would improve the model.