Programming for data science-LAB-EXP-8

Date-18-09-2025

Skewness Handling, Transformation, and Regression Evaluation (100 Marks)

Title: Skewness Handling, Transformation, and Regression Evaluation (100 Marks)

Design and implement an end-to-end data analysis workflow in R for a real-life regression problem: dataset creation with controlled missingness, skewness diagnosis, skewness reduction through transformations, visualization, and model performance comparison before vs after transformation.

Learning Outcomes

- Create synthetic datasets at scale with realistic distributions and controlled missingness.

- Diagnose and reduce data skewness using appropriate transformations.

- Build and evaluate regression models with standard metrics.

- Communicate findings with clear code, visuals, and interpretations.

Instructions to Students

1) Choose a real-life problem

- Select a domain such as healthcare, e-commerce, agriculture, finance, or transportation.

- Define a clear regression target (e.g., length of stay, order value, crop yield, loan amount, trip duration).

- Write a 3–5 sentence problem statement, including why skewness may naturally arise in this domain.

2) Create the dataset

- Programmatically generate a dataset with at least 2,000 rows and at least 10 attributes (mix of numeric and categorical allowed).

- Ensure that at least 3% of values are missing in selected attributes. Clearly document which attributes, approximate percentage per attribute, and the assumed missingness mechanism (e.g., MCAR/MAR).

- Intentionally include realistic skew in several numeric variables (e.g., right-skewed expenditures or durations).

3) Handle missing values

- Show pre-imputation missingness summary (counts/percentages per attribute).

- Apply appropriate handling techniques (e.g., mean/median for numeric, mode for categorical, or model-based imputation). Justify choices based on distribution and domain.

- Show post-imputation checks confirming no unexpected distortions or data leakage in the modeling split.

4) Diagnose skewness

- Compute skewness for every numeric attribute using a consistent method. Report a table with attribute name, skewness value, and skewness direction.

- Visualize each numeric attribute before transformation using:

  - One distribution plot (histogram or density), and

  - One boxplot to show outliers and tail behavior.

- Briefly interpret the direction/magnitude of skewness and whether it may affect linear modeling assumptions.

5) Reduce skewness

- Apply at least two transformation families across skewed variables. Options include:

  - Log / log1p for non-negative variables,

  - Square root / cube root,

  - Box-Cox (positive only),

  - Yeo-Johnson (supports zero/negatives),

  - Quantile/rank-based normalization if justified.

- Recompute skewness after transformation and re-plot the same diagnostics. Summarize improvements and any trade-offs (interpretability, zero/negative handling, stability).

6) Modeling and evaluation

- Use a proper validation strategy (train/test split or resampling). Apply the same model class to:
  - Original dataset (with missing values handled).
  - Transformed dataset (post skewness reduction).
- Recommended baseline: linear regression. Alternatives are acceptable with justification, but use the same model class for a fair comparison.
- Report at least two metrics: RMSE, MAE, and optionally R^2/Adjusted R^2. Present results in a comparison table and discuss differences.

7) Deliverables

- **Submission format: A single Lab Report including R Script.**
- The report must include:
  - Problem statement and dataset design rationale.
  - Missingness plan, handling method, and diagnostics (pre/post).
  - Skewness table and visualizations before/after transformations.
  - Modeling setup, metric tables for both datasets, and interpretation of differences.
  - Reproducibility details (set.seed and session info/package versions).
- Code must be well-commented and runnable end-to-end.

Marking Rubric (100 Marks)

A) Dataset creation and realism — 10

- Meets scale: 2,000+ rows and 10+ attributes.
- Realistic attribute design with clear domain rationale.
- Intentional inclusion of skew in selected numeric variables.

B) Missing values: injection and handling — 10

- At least 3% missing values in specified attributes; pattern documented.
- Appropriate, justified handling; clear pre/post missingness diagnostics.

C) Skewness computation and reporting — 15

- Correct, consistent skewness method used and reported for all numeric attributes.

- Clear table with numeric skewness values and direction.

D) Visual diagnostics: before and after — 15

- Appropriate histograms/density plots and boxplots for each numeric attribute pre- and post-transformation.

- Coherent commentary on direction/magnitude and outliers.

E) Transformations and rationale — 15

- At least two transformation families used, matched to data constraints.

- Quantitative reduction in skewness demonstrated; trade-offs and interpretability discussed.

F) Modeling on both datasets — 20

- Proper validation (split or resampling). Same model class on original vs transformed datasets.

- Clean preprocessing flow, no data leakage, reproducible code.

G) Performance comparison with metrics — 10

- At least two metrics (RMSE, MAE, optional R^2/Adj R^2) correctly computed.

- Comparison table with thoughtful interpretation of changes.

H) Code quality and documentation — 5

- Clean, modular, well-commented code; set.seed; readable structure.

I) Reporting clarity — 5

- Concise, well-structured write-up integrating figures and tables; limitations and next steps.

Required Structure in Word

- Title page: course, lab title, student name/Reg.No, date.

- Sections: Problem Statement; Data Generation; Missingness; Skewness Diagnosis; Transformations; Modeling; Results & Discussion; Limitations & Future Work; Appendix (Code, Session Info).

- Tables: Skewness summary; Model metrics comparison; Missingness summary.

- Figures: For each numeric attribute, before/after distribution plot and boxplot (may be arranged as panels).

File Naming

- Lab_Skewness_Transformation_Regression_StudentName_ID.docx

Important Notes:

Normalization Methods: Yeo–Johnson and Quantile/Rank- based Normalization

This document explains how to compute and when to use two common normalization techniques:

- Yeo–Johnson transformation (handles zero and negative values)
- Quantile / rank- based normalization (and when it is justified)

1. Yeo–Johnson Transformation

Overview:

The Yeo–Johnson transformation is a power transformation designed to stabilize variance and make data more normally distributed. Unlike the Box–Cox transform, Yeo–Johnson can be applied to zero and negative values.

Mathematical definition (for a single observation x and parameter λ):

If $x >= 0$:
  $T_\lambda(x) = ((x + 1)^\lambda - 1) / \lambda$ if $\lambda \neq 0$
  $T_\lambda(x) = \log(x + 1)$ if $\lambda = 0$

If $x < 0$:
  $T_\lambda(x) = - [(-x + 1)^{(2 - \lambda)} - 1] / (2 - \lambda)$ if $\lambda \neq 2$

T_λ(x) = -log(-x + 1)   if λ = 2

Notes: λ (lambda) is typically estimated from the data by maximum likelihood: choose λ that maximizes the likelihood under the assumption that the transformed data are normally distributed.

Step- by- step calculation (manual)

1. Choose a grid of candidate λ values (e.g., from -5 to 5).

2. For each λ, transform all observations x_i using the formula above.

3. For the transformed dataset, compute the log- likelihood under a normal model:

   LL(λ) = - (n/2) * log(σ_λ^2) + (λ - 1) * sum(I[x_i >= 0] * log(x_i + 1) - I[x_i < 0] * log(-x_i + 1))

   where σ_λ^2 is the sample variance of transformed values and I[...] is the indicator function. (In practice libraries compute this numerically.)

4. Choose λ that maximizes LL(λ). If you want a quick approach, use built- in routines in R (examples below).

R example (bestNormalize)

```r
library(bestNormalize)
X <- c(-2, -1, 0, 1, 2, 10)
res <- yeojohnson(X)
X_t <- predict(res, X)
# res$lambda gives estimated lambda
```

When to use Yeo–Johnson

Use when your variable is skewed and includes zero or negative values and your goal is to stabilize variance or make data closer to normality for modeling assumptions (e.g., linear regression, ANOVA).

Caveats

- The transformation is monotonic but not linear — interpretability of transformed coefficients requires care.
- Always check diagnostics (histogram, Q–Q plot) before and after.
- If modeling algorithms are scale- invariant (tree- based models), transformation may be unnecessary.

2. Quantile / Rank- based Normalization

Overview:

Quantile (rank- based) normalization maps the distribution of a variable to a target distribution (commonly uniform or normal) using ranks. It is non- parametric and useful when you want to remove distributional differences across samples or force normality regardless of original scale.

Two common variants

A. Quantile transform to uniform / normal (feature- wise):

1. Compute ranks r_i of observations x_i (average ranks for ties).
2. Convert ranks to uniform quantiles $u_i = r_i / (n + 1)$ (or r_i / n).
3. If you want normal scores, apply inverse CDF of target (e.g., $\Phi^{-1}(u_i)$ for standard normal).

B. Quantile normalization across samples (microarray style):

Used when you have multiple samples/features and want them to share the same distribution (e.g., expression microarrays):
1. Sort each sample's values.
2. Replace each sorted value by the row- wise average across samples.
3. Reassign to original positions so each sample has the common distribution.

Mathematical detail: rank- based inverse normal transform

Given ranks r_i (1..n), the transformed value $z_i = \Phi^{-1}((r_i - a) / (n + 1 - 2a))$ where a is a constant (Blom $a=3/8$, others use $a=0.5$ for $(r-0.5)/n$). Common choices:
- Van der Waerden: use $(r - 0.5)/n$
- Blom: $(r - 3/8)/(n + 1/4)$

These map ranks to quantiles and then to normal scores using the inverse normal CDF Φ^{-1}.

When is quantile/rank- based normalization justified?

Justifications:
- You need strict distributional equality across groups or features (e.g., combining batches with different distributions).
- Downstream methods assume normality and are sensitive to tails/outliers; rank- based transforms provide a nonparametric way to reduce heavy tails.

Not justified or use with caution:
- If absolute differences/relative magnitudes carry domain meaning (e.g., income, concentrations) — rank mapping destroys scale and relative distances.
- If downstream model interprets feature magnitudes directly (coefficients), interpretation becomes difficult.
- When preserving monotonic relationships and not exact distances is acceptable, rank transformations are fine.

Step- by- step calculation (rank- based inverse normal)

1. Compute ranks r_i (use average ranks for ties).

2. Compute fractional ranks $u_i = (r_i - 0.5) / n$ (or other formula as desired).

3. Compute $z_i = \Phi^{-1}(u_i)$ where Φ^{-1} is the inverse standard normal CDF.

4. Optionally standardize z_i to zero mean and unit variance (though Φ^{-1} of uniform quantiles already has theoretical mean 0).

R example (rank inverse normal)

```r
x <- c(10,20,30,1000)
r <- rank(x, ties.method='average')
```

```
u <- (r - 0.5)/length(x)
library(qnorm) # base stats has qnorm
z <- qnorm(u)
```

When to prefer Yeo–Johnson vs. Quantile normalization

- Use Yeo–Johnson when you want a parametric, monotonic power transform that stabilizes variance and often improves normality while preserving relative distances approximately.
- Use quantile/rank- based normalization when you need to enforce a specific distribution (e.g., strict normality) or remove distributional differences across groups — but accept that exact magnitudes and spacing are lost.
- If interpretability of transformed values matters (e.g., effect sizes), Yeo–Johnson is usually preferable.

Practical checklist

- Visualize before and after (histogram, boxplot, Q–Q plot).

- Check for ties and outliers; rank transforms handle outliers by compressing them into tails.

- Decide whether you need to preserve relative distances (Yeo–Johnson) or only ranks (quantile).

- If using quantile normalization across samples, make sure biological/real differences are not being erased.

Appendix: References & further reading

- Yeo, I.-K. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry.
- Peters, A., & others: Tutorials on quantile normalization and rank transformations.

3. Box–Cox Transformation

Overview:

The Box–Cox transformation is a family of power transformations introduced by Box and Cox (1964). It aims to stabilize variance and make data closer to normally distributed, improving the fit of statistical models. Unlike Yeo–Johnson, Box–Cox requires strictly positive values ($x > 0$).

Mathematical definition

For an observation $x > 0$ and parameter λ:

$T_\lambda(x) = (x^\lambda - 1) / \lambda \qquad \text{if } \lambda \neq 0$
$T_\lambda(x) = \log(x) \qquad \text{if } \lambda = 0$

Here, λ is chosen to maximize the log- likelihood of transformed data under a normal distribution assumption.

Step- by- step calculation

1. Ensure all values are strictly positive. If not, add a constant shift to make them positive.

2. Select candidate λ values (e.g., −5 to 5).

3. Transform data using the Box–Cox formula for each λ.

4. Compute log- likelihood under a normal model; choose λ maximizing this.

5. Use transformed data for further modeling.

R example (MASS)

```r
library(MASS)
x <- c(1,2,3,4,5,10)
res <- boxcox(lm(x ~ 1))
# Shows profile likelihood for λ
```

When to use Box–Cox

- When your data are strictly positive and skewed.
- When variance stabilization and approximate normality are desired for parametric models.
- Simpler than Yeo–Johnson if no zero/negative values are present.

Comparison with Yeo–Johnson

- Box–Cox requires strictly positive values; Yeo–Johnson supports zero/negative.
- Both select λ to maximize likelihood.
- Interpretation of λ is similar: λ=1 leaves data nearly unchanged, λ=0 ~ log transform.
- Yeo–Johnson is more flexible, but Box–Cox can be more efficient when data are positive.