BAN404 Statistical Learning - R Cheat Sheet

Based on ISLR 2nd Ed. Spring 2025 Syllabus

Contents

1 Core Concepts & Workflow (Ch 1-2)

1.1 Statistical Learning Fundamentals

- **Goal**: Learn a function f relating predictors $X = (X_1, ..., X_p)$ to a response Y, typically modeled as $Y = f(X) + \epsilon$, where ϵ is mean-zero error.
- **Prediction**: Estimate Y using $\hat{Y} = \hat{f}(X)$. Accuracy is primary goal. $\hat{f}$ can be a black box.
- **Inference**: Understand the relationship between X and Y. Interpretability is primary goal. How does Y change as X_j changes? Which X_j are important? Is the relationship linear?
- **Supervised**: Both X and Y observed. Includes Regression (quantitative Y) and Classification (qualitative Y).
- **Unsupervised**: Only X observed. Find structure, e.g., PCA, Clustering.

1.2 Model Accuracy

- **Quality of Fit**: How well do predictions match observed data?
- **Training Error**: Calculated on the data used to fit the model. Usually lower than test error; can be overly optimistic.
- **Test Error**: Average error on new, unseen data. The true measure of predictive performance. Estimated using validation set or cross-validation.
- **Measures (Regression)**: Mean Squared Error (MSE) $= \frac{1}{n}\sum(y_i - \hat{f}(x_i))^2$. Residual Standard Error (RSE) $= \sqrt{RSS/(n-p-1)}$. R^2 = Proportion of variance explained.
- **Measures (Classification)**: Classification Error Rate $= \frac{1}{n}\sum I(y_i \neq \hat{y}_i)$. Confusion Matrix, Sensitivity, Specificity, ROC Curve, AUC.

1.3 Bias-Variance Trade-Off (Sec 2.2.2)

- For a given test point x_0, $E(\text{Test MSE at } x_0) = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon)$
- **Variance**: Amount $\hat{f}$ would change if fit on a different training set. More flexible methods have higher variance.
- **Bias**: Error introduced by approximating a complex real-life problem with a simpler model. More flexible methods have lower bias.
- **Irreducible Error** ($\text{Var}(\epsilon)$): Cannot be reduced by model choice.
- **Trade-off**: Flexible models $\downarrow$ bias, $\uparrow$ variance. Inflexible models $\uparrow$ bias, $\downarrow$ variance. Goal is to minimize Test MSE, often achieved at intermediate flexibility. Test error typically shows a U-shape vs. model flexibility.

1.4 K-Nearest Neighbors (KNN) Intro (Sec 2.2.3)

- Non-parametric method for classification/regression.
- **Classification**: Predict class based on majority vote of K nearest training observations. Decision boundary can be highly non-linear for small K.
- **Regression**: Predict response by averaging responses of K nearest training observations.
- **Choice of K**: Controls flexibility. Small K = low bias, high variance (wiggly fit). Large K = high bias, low variance (smooth fit). Use CV to choose K.
- Requires scaling predictors. Suffers from curse of dimensionality (needs $n \gg p$).

2 Basic R Operations & Data Handling

- **Core Functions**: `c()`, `matrix()`, `data.frame()`, `library()`, `?()`, `ls()`, `rm()`, `names()`, `dim()`, `head()`, `summary()`, `str()`
- **Stats**: `mean()`, `var()`, `sd()`, `cor()`, `quantile()`, `table()`
- **Missing Data**: `is.na()`, `sum(is.na())()`, `na.omit()`
- **Subsetting**: `[rows, cols]`, , *logicals*, -(*omit*)`Reading Data` : `read.csv()`, `read.table()`(*use*`header=T`, `na.strings`)
- **Scaling**: `scale()`
- **Factors**: `as.factor()`, `contrasts()`, `relevel()`
- **Dummy Vars**: `model.matrix()`
- **Splitting Data**: `set.seed()`, `sample()`
- **Plotting**: `plot()`, `hist()`, `boxplot()`, `pairs()`, `abline()`, `points()`, `lines()`, `legend()`, `par(mfrow=...)()`
- **Apply Functions**: `apply(X, MARGIN, FUN)()` (MARGIN=1 for rows, 2 for columns)
- **Writing Functions**:

```
my_function <- function(arg1, arg2 = default_value) {
  # Computations...
  result <- ...
  return(result)
}
```

3 Linear Regression (Ch 3, Lab 3.6)

- **Concept**: Models Y as a linear combination of predictors X_j. $Y = \beta_0 + \sum \beta_j X_j + \epsilon$.
- **Fitting**: `lm()` minimizes RSS.

```
fit <- lm(y ~ x1 + x2 * x3, data=mydata, subset=train_indices) # Includes interaction
fit_poly <- lm(y ~ poly(x1, 3), data=mydata) # Degree 3 polynomial
```

- **Interpretation**: `summary()` gives key stats. β_j is avg. change in Y for one unit change in X_j, holding others constant.
- **Hypothesis Tests**:
  - t-test: Is $\beta_j = 0$? (p-value in `summary()`)
  - F-test: Are all $\beta_j = 0$? (F-statistic in `summary()`)
- **Confidence Interval**: Range for true parameter value. `confint()`.
- **Prediction Interval**: Range for a single future observation. Wider than CI. `predict()` with `interval="prediction"`.
- **Diagnostics**: Use `plot(fit)()` to check assumptions (linearity, constant variance, normality of errors) and identify outliers/leverage points. Use `vif()` (**car**) for multicollinearity.

4 Classification (Ch 4, Lab 4.7)

- **Goal**: Predict categorical Y.
- **Logistic Regression**: Models $P(Y = k|X)$ using logit link.
  - Fit: `glm()` with `family=binomial`.
  - Predict probabilities: `predict(fit, type="response")`.
  - Thresholding: Convert probabilities to class predictions (e.g., threshold 0.5).
  - Interpretation: Coefficients represent change in log-odds.
- **LDA**: (**MASS**) Assumes $X|Y = k \sim N(\mu_k, \Sigma)$. Linear boundary. Robust, good for small n, stable if classes separated. `lda()`.
- **QDA**: (**MASS**) Assumes $X|Y = k \sim N(\mu_k, \Sigma_k)$. Quadratic boundary. More flexible, needs more data. `qda()`.
- **Naive Bayes**: (**e1071**) Assumes predictors conditionally independent within class. Good for high p. `naiveBayes()`.
- **KNN**: (**class**) Non-parametric. Majority vote of K neighbors. Needs scaling. `knn()`.
- **Evaluation**: Confusion Matrix (`table()`), Accuracy (`mean()`), ROC/AUC (**ROCR**). Changing threshold (e.g., from 0.5 to 0.2) affects sensitivity/specificity trade-off.

5 Resampling Methods (Ch 5, Lab 5.3)

- **Cross-Validation (CV)**: Estimates test error.
  - Validation Set: Simple split, variable results.
  - LOOCV: $k = n$. Unbiased but high variance. Use `cv.glm()` (**boot**).
  - k-Fold CV: $k = 5$ or 10 common. Good bias-variance balance. Use `cv.glm()`, `cv.tree()`, `cv.glmnet()`, `tune()`, or manual loop. *Remember to perform model selection steps within each fold if tuning.*
- **Bootstrap**: Resample data *with replacement* B times. Estimate standard error / CIs for statistics without relying on formulas/assumptions. Use `boot()` (**boot**). Define a function to calculate the statistic of interest on a sample specified by indices.

6 Linear Model Selection and Regularization (Ch 6, Lab 6.5)

- **Motivation**: Reduce variance, improve prediction, enhance interpretability when p is large or $p \approx n$.
- **Subset Selection**: (**leaps**) `regsubsets()`
  - Best Subset: Evaluates all 2^p models. Use Cp, BIC, Adj R^2, CV error to choose best size.
  - Stepwise (Forward/Backward): Greedy search. Computationally faster.
- **Shrinkage**: (**glmnet**) Penalizes large coefficients.
  - Ridge (`alpha=0`): L2 penalty ($\lambda \sum \beta_j^2$). Includes all variables, shrinks towards zero. Good for collinearity.
  - Lasso (`alpha=1`): L1 penalty ($\lambda \sum |\beta_j|$). Performs variable selection (sets some $\beta_j = 0$). Sparse models.

- Tuning λ: Use `cv.glmnet()`. `$lambda.min`, `$lambda.1se`.
- Data prep: Use `model.matrix()` for X, standardize usually recommended.

- **Dimension Reduction**: (**pls**) Create $M < p$ components Z_m.
  - PCR: Uses principal components (unsupervised). `pcr()`. Tune `ncomp`.
  - PLS: Components derived using Y (supervised). `plsr()`. Tune `ncomp`.
  - Use `scale=TRUE`, `validation="CV"`. `validationplot()` to choose M.
- **High Dimensions**: Focus on Ridge, Lasso, PCR, PLS. Evaluate using Test/CV error. Training error (R^2, RSS) is meaningless. Interpretation requires care due to extreme collinearity.

7 Moving Beyond Linearity (Ch 7, Lab 7.8)

- **Polynomials**: `poly(X, degree=d)`, `I(Xd̂)`. Simple but can be unstable.
- **Step Functions**: `cut()`. Piecewise constant.
- **Regression Splines**: (**splines**) `bs()`, `ns()`. Piecewise polynomials joined smoothly at knots. `df` controls flexibility. Natural splines (`ns()`) are linear beyond boundaries.
- **Smoothing Splines**: `smooth.spline()`. Uses penalty on second derivative for smoothness. λ or `df` controls smoothness. `cv=TRUE` finds λ via LOOCV.
- **Local Regression**: `loess()`. Weighted regression in local neighborhoods. ‘span‘ controls neighborhood size/smoothness.
- **GAMs**: (**gam**) Extends linear/logistic models: $g(E[Y]) = \beta_0 + \sum f_j(X_j)$. Uses `s()` (smoothing spline) or `lo()` (loess) terms. Fit additively via backfitting. Check non-linearity with `anova()`. Use `family=binomial` for classification.

8 Tree-Based Methods (Ch 8, Lab 8.3)

- **Decision Trees**: (**tree**) Recursive binary splitting. Prone to overfitting. Prune using CV (`cv.tree()`, `prune.tree()`). Easy interpretation.
- **Bagging**: (**randomForest**) Bootstrap aggregation. Average B trees fit on bootstrap samples. Reduces variance. Set `mtry=p`.
- **Random Forests**: (**randomForest**) Bagging + feature randomness (`mtry < p`). Decorrelates trees, often improves over bagging. `importance()`, `varImpPlot()`.
- **Boosting**: (**gbm**) Sequential fitting on residuals. Slow learning via shrinkage (λ). Can overfit. Tune `n.trees`, `shrinkage`, `interaction.depth`. Partial dependence plots.
- **BART**: (**BART**) Bayesian approach, ensemble of trees via MCMC perturbation. Often strong performance with minimal tuning.

9 Support Vector Machines (Ch 9, Lab 9.6)

- **Hyperplane**: Separates p-dimensional space. Defined by $\beta_0 + \sum \beta_j X_j = 0$.
- **Maximal Margin Classifier**: Largest margin separating hyperplane for separable data.
- **Support Vector Classifier (SVC)**: Linear boundary, uses soft margin allowing violations (ϵ_i) controlled by `cost` (C). Finds max margin subject to budget C for violations. Uses `kernel="linear"` in `svm()` (**e1071**).

- **Support Vector Machine (SVM)**: Uses kernels for non-linear boundaries.
  - Polynomial: `kernel="polynomial"`, tune `degree`, `cost`.
  - Radial: `kernel="radial"`, tune `gamma`, `cost`.
- **Support Vectors**: Points on or violating the margin (influence the boundary).
- **Tuning**: Use `tune()` (**e1071**) with CV to select kernel parameters (`cost`, `gamma`, `degree`).
- **Multi-class**: Handled via one-vs-one or one-vs-all. **e1071** uses one-vs-one.

10 Unsupervised Learning (Ch 12, Lab 12.5)

- **Goal**: Discover structure in X only (no Y).
- **Principal Components Analysis (PCA)**: Find low-dimensional linear combinations (PCs) capturing maximum variance. Used for visualization and dimension reduction.
  - R: `prcomp()` (`scale.=TRUE` recommended). `$x` are scores, `$rotation` are loadings.
  - PVE: Proportion of Variance Explained. Use `summary()` or plot `pr.out$sdev^2 / sum(pr.out$sdev^2)`. Look for elbow in scree plot.
- **Matrix Completion**: Impute missing values, e.g., using iterative SVD (Alg 12.1).
- **Clustering**: Partition observations into groups (clusters).
  - **K-Means**: Partition into K pre-specified clusters minimizing within-cluster variance. R: `kmeans()` (`centers=K`, `nstart=25`). Sensitive to initial assignment and scaling. Need to choose K.
  - **Hierarchical**: Builds a dendrogram (tree). No need to pre-specify K.
    - Dissimilarity: Euclidean (`dist()`), correlation (`as.dist(1-cor(t(data)))`).
    - Linkage: `method="complete"`, `"average"`, `"single"`, `"centroid"` in `hclust()`.
    - Cut tree: `cutree()`.
- **Considerations**: Scaling, choice of distance/linkage, choice of K are important practical decisions.