**0) 환경 & 산출물**

* **주요 패키지:** Seurat/SingleCellExperiment+scran/scater, glmnet, e1071 또는 kernlab, rsample(또는 caret/tidymodels), irlba(PCA), pls(PLS), *(선택)* ontologyIndex/igraph(CO), pROC/yardstick.
* **아티팩트:** 전처리 객체(Seurat/SCE), **fold 스플릿**(재현성), **Level-0 OOF 예측 테이블**, **Level-1 메타 모델**, **평가 리포트**(leaf + hierarchical metrics), *(선택)* CO 거리행렬 CSV.

**1) 데이터 분할 & 기본 전처리**

1. **Train/Test 분리** (고정 시드): rsample::initial\_split.
2. **Train 내부 3-fold** 생성: vfold\_cv(v=3)  
   → **모든 마커 선택, 임베딩 학습, Level-0 학습은 fold-내에서만** 수행(누수 방지).
3. **QC/정규화/특징 선택**
   * QC: 유전자·UMI·미토 비율 필터.
   * 정규화: SCTransform 또는 NormalizeData+ScaleData (배치가 크면 Harmony/batchelor 고려).
   * HVG 선정: FindVariableFeatures 또는 scran::modelGeneVar.

산출: train\_obj, test\_obj, fold 인덱스.

**2) Level-0 모델군 (두 갈래 병렬)**

**A) 1-vs-rest (cell-type specific marker 기반)**

* **마커 선택(각 fold의 train만 사용):**
  + 간단: Seurat::FindAllMarkers(Wilcoxon)에서 **타입별 top-200**.
  + 질문의 방식: **chi-square (presence/absence)** → CPM/CP10K 기준 이진화 후 chisq.test로 각 타입 vs rest, 상위 200.
* **모형 학습:**
  + 각 타입 ttt: glmnet::cv.glmnet(family="binomial", alpha=0, standardize=TRUE)
  + (대안) kernlab::ksvm 또는 e1071::svm(prob=TRUE).
* **예측 산출(OoF):**
  + fold-holdout에 대해 **확률**(또는 로짓) 추출 → pred\_LR\_1vR[, t], pred\_SVM\_1vR[, t].

**B) 전역 차원축소 임베딩 → 멀티클래스 분류**

* **임베딩 학습(각 fold의 train만):**
  + PCA: irlba::prcomp\_irlba(상위 K, 예: 50).
  + (권장) **PLS**: pls::plsr(Y=one-hot, 감독형 축소) → 분류 관련 축 보존.
* **모형 학습:**
  + glmnet::cv.glmnet(family="multinomial", alpha=0)
  + kernlab::ksvm(..., type="C-svc", prob.model=TRUE)
* **예측(OoF):**
  + fold-holdout에 대해 클래스별 확률 TTT개씩 → pred\_LR\_multi[, 1..T], pred\_SVM\_multi[, 1..T].
  + *(옵션)* **임베딩 좌표**(K차원)도 저장 → Level-1에 추가 입력(S2 옵션).

산출: 각 fold에서 **2T(or 4T)개**의 OOF 예측 + *(옵션)* 임베딩 K차원.  
주의: **확률 교정**(Platt/Isotonic)은 메타 안정성에 도움. 기본은 ridge-LR가 교정 효과를 일부 제공.

**3) Level-1 스태킹(메타) 단계**

**입력 구성**

* **S1 (기본):** [pred\_LR\_1vR,pred\_SVM\_1vR,pred\_LR\_multi,pred\_SVM\_multi][pred\\_LR\\_1vR, pred\\_SVM\\_1vR, pred\\_LR\\_multi, pred\\_SVM\\_multi][pred\_LR\_1vR,pred\_SVM\_1vR,pred\_LR\_multi,pred\_SVM\_multi] ⇒ **2T** 또는 **4T** 열.
* **S2 (확장):** S1 + **임베딩 K차원**(overfit 방지 위해 K를 10–30으로 축소하거나 ridge 강하게).

**메타 러너**

* 기본: glmnet::cv.glmnet(family="multinomial", alpha=0) (ridge).
* **손실 선택:**
  + **표준 CE**: 그대로 multinomial.
  + **계층형 손실(스태킹 단계만)**: 구현이 번거롭다면 **훈련은 CE**, **추론 시 비용-민감 의사결정**으로 대체해도 실용적.
    - 추론 규칙: y^=arg⁡min⁡c∑kCc,k pk\hat{y} = \arg\min\_c \sum\_k C\_{c,k}\, p\_ky^​=argminc​∑k​Cc,k​pk​ (CO 기반 비용 CCC).
    - 또는 **HXE-유사 soft target**을 만들어(정답 주변 계통에 확률 분산) glmnet 학습에 주입하는 간접 방식(샘플 복제+가중치로 근사).

산출: **메타 모델**(ridge-multinomial), **온톨로지 비용행렬**(선택), 추론 규칙.

**4) 최종 학습(Full train) & 테스트 평가**

1. **Full train에서 Level-0 재학습**(마커/임베딩 포함 → 누수 없이 재적합).
2. **Level-1 재학습:** full train에서 **S1/S2 입력 재생성 → 메타 재적합**.
3. **테스트**: test 세트에 대해 Level-0 → 메타 입력 생성 → 메타 예측.
4. **계층형 후처리(선택):**
   * **DAG 일관성 보정**(isotonic-like): 조상≥자식 보정.
   * **비용-민감 결정**: 위 식 적용.
   * **Partial/Unknown 정책**: top-1 확률 < τ & 조상 확률 > τ ⇒ 상위 노드 반환; 매니폴드 거리 ↑ ⇒ unknown.

**지표**

* **leaf**: accuracy, macro-F1.
* **교정**: Brier, ECE.
* **계층형**: 평균 CO 거리(예측–정답), hierarchical P/R.
* **희귀/경계**: rare-class recall, near-miss 비율, unknown AUROC.

**5) 파일 구조(권장)**

bash

복사편집

project/

data/raw/

data/processed/

splits/ folds.rds

models/ level0\_marker/ level0\_embed/ level1\_meta/

preds/ oof/ test/

ontology/ cl\_dissimilarity.csv label\_map.csv

R/

00\_utils.R

10\_preprocess.R

20\_marker\_selection.R

30\_level0\_marker.R

31\_level0\_embed.R

40\_make\_oof.R

50\_level1\_meta.R

60\_predict.R

70\_metrics.R

**6) 아주 짧은 코드 스케치(핵심 흐름)**

r

복사편집

# Splits

set.seed(42)

library(rsample)

split <- initial\_split(meta\_df, prop=0.8, strata=cell\_type)

train\_ids <- training(split)$cell\_id; test\_ids <- testing(split)$cell\_id

vfold <- vfold\_cv(training(split), v = 3, strata = cell\_type)

# Level-0 (marker branch) inside each fold

for (fold in vfold$splits) {

tr <- analysis(fold); va <- assessment(fold)

markers <- select\_markers(tr, method="chisq", top\_n=200) # fold-internal

fit\_lr\_list <- fit\_one\_vs\_rest(tr, markers, model="glmnet") # T binomial models

fit\_svm\_list <- fit\_one\_vs\_rest(tr, markers, model="svm")

oof\_lr <- predict\_oof(va, fit\_lr\_list) # Nval x T

oof\_svm <- predict\_oof(va, fit\_svm\_list)

save\_oof(...)

}

# Level-0 (embedding branch)

for (fold in vfold$splits) {

tr <- analysis(fold); va <- assessment(fold)

emb <- fit\_embed(tr, method="pls", k=50) # fold-internal embedding

fit\_lr <- fit\_multiclass(tr, emb, model="glmnet") # multinomial

fit\_svm <- fit\_multiclass(tr, emb, model="svm")

oof\_lr <- predict\_multiclass(va, emb, fit\_lr) # probs Nval x T

oof\_svm <- predict\_multiclass(va, emb, fit\_svm)

save\_oof(...); save\_emb\_coord(...)

}

# Level-1 meta

X\_oof <- cbind(oof\_marker\_lr, oof\_marker\_svm, oof\_embed\_lr, oof\_embed\_svm, emb\_coord\_opt)

y\_oof <- factor(cell\_type\_oof)

meta <- glmnet::cv.glmnet(x = as.matrix(X\_oof), y = y\_oof,

family="multinomial", alpha=0) # ridge

# Predict on test

X\_test <- make\_meta\_inputs\_on\_test(...)

p\_meta <- predict(meta, as.matrix(X\_test), type="response")[,,1] # N x T

# (optional) cost-sensitive decision using CO distance

y\_hat <- apply(p\_meta %\*% t(cost\_matrix), 1, which.min) # or custom function

**7) 선택 옵션(빠르게 성능 올리는 팁)**

* \*\*PLS(감독형 축소)\*\*를 우선 적용(분류 신호를 보존).
* 메타 입력은 **확률 대신 로짓**을 쓰면 가끔 안정적.
* 메타에서 **ridge(α=0)** 기본, 필요시 elastic-net(α∈[0,0.3]) 탐색.
* 임베딩을 메타에 넣을 때는 **차원 축소** 또는 **강한 정규화** 필수.
* **계층형 후처리**(일관성 보정 + 비용-민감 결정)만으로도 “near-miss 유도” 효과 큼.