学　　号：20183744

天 津 商 业 大 学 毕 业 论 文

线性判别分析的原理与应用

Linear Discriminant Analysis Theory & Application

学　　院:　　理学院

教 学 系:　　统计系

专业班级:　　统计学专业 1801

学生姓名:　　刘士坤

指导教师:　　王倩　　讲师

2022 年 5 月 20 日

目　　录

摘要

本文首先对线性判别分析的发展及其理论作了一个概述，从最简单直观的距离判别法到 Fisher 的最优线性判别函数，再到后面的 Bayes 判别规则。这三种方法在两总体正态同方差下导出判别函数的一致性也从不同的角度证明了这些判别方法的合理性。

在线性判别分析的应用上，本文对英雄联盟 14 分钟时对局的时间线数据建立线性判别分析（主要是 Bayes 判别）模型。在建模前，先对数据做了探索性数据分析，并分析了各变量对蓝色方与紫色方获胜胜率的影响，得出了控小龙比控峡谷先锋更好赢以及上单的 Carry 能力不如中单和 ADC，这些结论在平时游戏里都或多或少能感受得到。建立的 Bayes 判别函数的预测正确率为 75%，由于我们在模型中只考虑了小龙数、峡谷先锋数、防御塔丢失数以及各位置的对位经济差，没有考虑双方的英雄选择及阵容，而这对整局游戏的影响无疑是非常重要的，所以 75%可以接受。最后通过对误判样品和正判样品的比较，发现误判样品大多为双方团队的经济及资源获取在 14 分钟时没有拉开比较大的差距，还有一些样品由于阵容及某些英雄前期弱势后期强势，而这些模型并没有考虑，从而导致误判。

关键词：线性判别分析；Bayes 判别；英雄联盟单双排对局获胜预测

Abstract

This article first introduces the development and theory of linear discriminant analysis, from the simplest and intuitive distance discriminant method to Fisher's best linear discriminant function, and later to Bayesian discriminant rule. The consistency of the discriminant functions derived by these three methods when two multivariate normal populations with the same covariance matrix also proves the rationality of these methods from different perspectives.

This article develops a linear discriminant analysis(mainly Bayesian discriminant) model for the timeline data of league of legends at 14:00 as an application for linear discriminant analysis. Before modeling, exploratory data analysis on the data is done and analyze the impact of each variable on the winning percentage of the blue and red side and came to the conclusion that killing the dragon is better than the rift herald to win as well as the Top Laner is hard to carry this game by himself, but the Mid Laner and AD Carry are the opposite. That is why it is called AD Carry. These conclusions can be felt more or less in our usual game. The prediction accuracy of the established Bayesian discriminant function is 75%. Since we only consider dragon kills@14, rift herald kills@14, towers taken@14 by lane and the gold difference@14 by role, and did not consider both sides of the champion picks, which is undoubtedly important to this game. So 75% is acceptable. Finally, comparing the misclassfied samples with the correct samples, it was found that most of the misclassfied samples were the gold and objectives taken of both teams without a relatively large gap. And there were also small samples due to the champion picks that some champions are weak in the early stage of this game and strong in the late stage of this game. But these factors were not contained in the model, thus leading to misclassfied.

Key Words: Linear Discriminant Analysis, Bayesian Discriminant Rule, The Prediction of Game Winner for League of Legends Solo/Duo Rank.

1 引言

1.1 研究背景及意义

1.1.1 判别分析的由来

我们在生产生活中经常会遇到如何根据观测到的数据资料对所研究的对象进行判别归类的问题。例如：在新冠肺炎确诊病例的诊断中，一个病人有发烧、咳嗽、头痛、呼吸困难等症状，医生要判断他患的是普通流行性感冒还是新型冠状病毒。在气象预测中，根据过去已有气象资料（如气温、气压、湿度等）来判断明天是晴天还是阴天，进而判断是有雨还是无雨。除了这些以外，社会调查、经济学、互联网等领域都存在判别问题，可以说判别分析发展到今天已经渗透到人类社会的各个领域。但不管是哪个领域，判别分析问题一般都可以这样描述：设有 k 个 p 维总体 $\boldsymbol{G}_1, \boldsymbol{G}_2, \cdots, \boldsymbol{G}_k$，其总体分布特征已知（如已知分布函数分别为 $\boldsymbol{F}_1(x), \boldsymbol{F}_2(x), \cdots, \boldsymbol{F}_k(x)$，或有来自各个总体的样本，从这 k 个总体中抽样来进行推断）。对于给定的一个新样品 $\boldsymbol{X}$，我们要判断它来自这 k 个总体中的哪一个[1][2]。

1.1.2 线性判别分析的提出

在数理统计学上判别问题的实质就是认为样本的各种指标数据来自于不同的分布，然后分别估计其中的参数，获得关于总体的统计特征，以此来进行判别分类。基于此，在进行判别归类时根据所要研究问题的条件、判别时的依据从而可以得到各种不同的判别方法，本文论述的线性判别分析就是其中一种。

线性判别分析是 R. A. Fisher 在其 1936 年的经典论文 The use of multiple measurements in taxonomic problems[3]中提出的，论文中同时也公布了现如今经常出现在统计学习或机器学习中的一个示例数据集——鸢尾花数据集。由于鸢尾花测量数据的特征和样本量都比较少，并且不同品种的鸢尾花组间差异非常明显，故而经常

在教学中被用来做判别或聚类（删掉鸢尾花“品种”这个特征变量）。Fisher 就是对这个数据集做了一个线性判别分析，但其目的并不是为了对三种鸢尾花进行判别归类，而是为了计算三个品种之间亲缘关系的远近，在野外三种鸢尾花的真实判别依据是种子[4]。关于这一点 Fisher 在其论文中明确提到了，对鸢尾花的测量数据进行统计分析是为了说明 Iris versicolor 是 Iris virginica 与 Iris setosa 的中间品种，而且 Iris versicolor 和 Iris virginica 的距离与和 Iris setosa 的距离的比值应该是 1:2。

Fisher 所做的线性判别分析就是对四个特征变量做了一个线性变换，即以四个特征变量的加权和作为一个综合指标，以这个综合指标作为判别的依据，使其能更好地把这三个品种区分开，为此 Fisher 提出了一个准则：在判别函数总方差之上衡量判别函数在各个总体中的均值应该具有较大的离差。这个准则被称为 Fisher 准则。通过最大化 Fisher 准则可以求出最优的线性判别函数，通过比较这三个品种线性判别函数结果之间的距离就可以对其亲缘关系的远近进行推断。如**图 1** 所示，线性判别分析的结果证实了 1:2 这个比值。

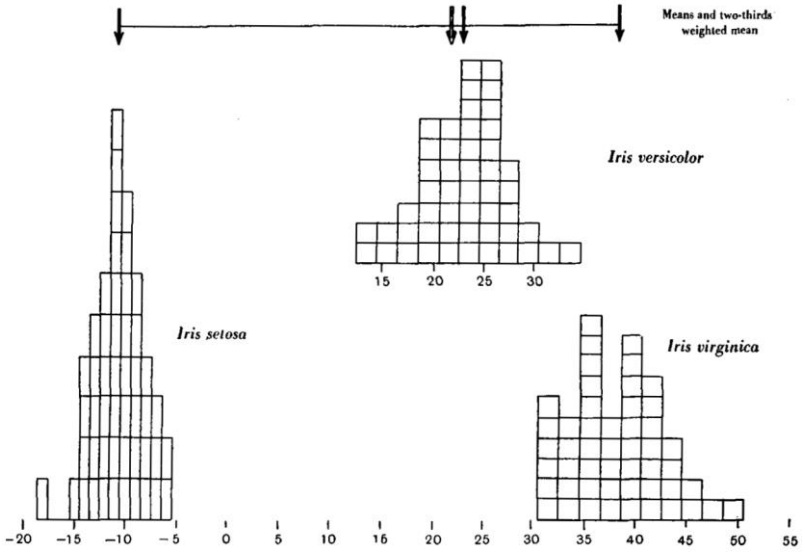

图 1 三种鸢尾花线性判别函数的频率直方图

资料来源：Fisher, R. A.. The use of multiple measurements in taxonomic problems [J]. Annals of eugenics. 1936, 7(2): 179-188.[3]

1.1.3 线性判别分析的一些成功应用

在生物医学上的应用如 1.1.2 所述，线性判别分析自 1936 年由 Fisher 首次提出并应用于生物分类。在医学领域通过历史患者资料建立判别函数有助于识别各种疾病的特征，并根据患者的临床症状将其分类。

在这之后，1968 年 Altman 将线性判别分析引入基于财务比率和其他金融变量的破产预测中并提出了 Altman Z-score 模型，是第一个用来系统解释公司进入破产或存活的统计学工具[5]。

80 年代后随着计算机的发展及机器学习和人工智能的兴起，处理的数据量越来越大，线性判别分析作为一种降维特征选择 **Error! Reference source not found.**的方法在 1996 年被 Belhumeur 引入模式识别和人工智能领域 **Error! Reference source not found.**，并广泛应用于语音识别 **Error! Reference source not found.**、人脸识别 **Error! Reference source not found.**、步态识别[10]、手势识别[11]、行人再识别[12]等。如在人脸识别系统中，人脸图片被转化为大量的像素，线性判别分析就可以在分类之前将这些像素转化为数量更少、更易于处理的特征，这些特征也因此被叫做 Fisher 脸。

本文则将线性判别分析应用到《英雄联盟》的对局时间线数据中，通过结合时间线数据的探索性分析及线性判别分析以期得到一些对英雄联盟玩家有帮助的信息，并希望利用游戏前期的数据，通过线性判别分析预测对局的胜负。

1.2 研究内容及方法

1.2.1 本文的研究内容

本文首先介绍了判别分析的由来及线性判别分析的提出，并给出了线性判别分析的一些成功应用，说明了本文的研究意义。然后介绍了本文的研究内容和研究方法。

第二部分是对线性判别分析的概述。给出了距离判别的两种导出方法，一种是直

观法；另一种是通过仿照似然比导出判别函数。而这个判别函数其实就是个二次型的差，和欧几里得空间中的距离是一致的。基于此 Fisher 提出了只考虑判别函数是线性的那种情况，找到最优的线性判别函数，当然这也可以从几何投影的角度来解释。然后介绍了 Bayes 判别，即直接将 Bayes 的思想应用到到判别问题上，计算出使得平均误判损失（风险函数）最小的那个判别函数，也称为 Bayes 解。

在这之后介绍了线性判别分析在英雄联盟单双排对局预测中的应用。其中首先介绍了《英雄联盟》这款游戏及游戏的内容，如召唤师峡谷地图、英雄类型及分线，然后对探索性数据分析和线性判别分析所用到的数据进行了一些说明。在探索性数据分析部分借助统计软件 R 对数据中的各变量的统计特征及对局胜率进行了分析。最后一部分在对数据验证模型的假设之后，建立线性判别分析（主要是 Bayes 判别）模型，预测对局的胜负。

最后一部分是小结与展望，对全文的内容做了一个总结，并指出了本文线性判别分析在英雄联盟单双排对局预测中的一些问题和不足之处。

1.2.2 本文的研究方法

通过拳头游戏 API（https://developer.riotgames.com/）获取到游戏对局数据并利用统计软件 R 做探索性数据分析，通过表格、图形来说明数据的一些特征。在对数据进行线性判别分析建模预测胜负之前，通过表格、图形对双方胜率的影响因素进行探究。利用 MASS 软件包中的 lda 函数对数据建立线性判别分析（主要是 Bayes 判别）模型并对预测结果进行分析。

2 线性判别分析的理论及方法

本部分首先概述判别问题的距离判别法，并从两个角度导出了距离判别法的判别规则。其中一个角度是根据直观判断，样品距离哪个总体近就判给哪个总体。另一个角度则是似然比检验，通过仿照似然比检验给出判别函数，进而给出判别规则。距离判别导出的判别函数可以是一次函数也可以是二次函数，在此基础上引出只考虑判别函数是线性的那种情形，即 Fisher 线性判别函数。然后联想到多元分析很多方法都在做降维，不同方法的区别只是那个较低维的空间。最后给出在实际中使用更多的 Bayes 判别规则。

2.1 距离判别法[1][2]

在 Fisher 提出线性判别分析之前，统计学上关于判别问题已经有了不少的解决方法，如最简单且直观的距离判别方法。距离判别可以从很多不同的角度导出，直观坦率地讲就是样品和哪个总体的距离最近，就把它判给哪个总体，在具体实施时根据所研究问题的情况会采用不同的“距离”定义。通常使用比较多的“距离”定义是1936 年由印度统计学家 Mahalanobis 提出的马氏距离，该距离定义如下

$$d(\boldsymbol{x},\boldsymbol{y}) \triangleq [(\boldsymbol{x}-\boldsymbol{y})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{y})]^{\frac{1}{2}}. \tag{1}$$

也记作 $d(\boldsymbol{x},\ \boldsymbol{y}) \triangleq ||\boldsymbol{x}-\boldsymbol{y}||_{\boldsymbol{\Sigma}}$，其中 $\boldsymbol{\Sigma}$ 称为权矩阵，一般为总体的协方差矩阵，若总体未知可以用样本的协方差矩阵估计。

定义样本 $\boldsymbol{y}$ 到总体 $\boldsymbol{G_i}$ 的距离为

$$d(\boldsymbol{y},\boldsymbol{G_i}) \triangleq ||\boldsymbol{y}-\boldsymbol{\mu_i}||_{\boldsymbol{\Sigma}}. \tag{2}$$

从而，距离判别直观描述的数学表达为

$$\begin{cases} \boldsymbol{y} \in \boldsymbol{G}_i, & d(\boldsymbol{y},\boldsymbol{G}_i) < d(\boldsymbol{y},\boldsymbol{G}_j), \forall j \neq i; \\ \text{待判}, & d(\boldsymbol{y},\boldsymbol{G}_i) = d(\boldsymbol{y},\boldsymbol{G}_j), \exists j \neq i. \end{cases} \tag{3}$$

这就是距离判别的判别规则，即若 $\boldsymbol{y}$ 距离总体 $\boldsymbol{G_i}$ 最近，就判 $\boldsymbol{y}$ 属于 $\boldsymbol{G_i}$.

除了直观的距离导出以外，距离判别还可以从假设检验的角度导出，在正态总体同协方差矩阵的情形下，通过仿照似然比检验的想法可以导出两总体时的判别函数

$$W(\boldsymbol{y}) = ln\frac{f_1(y)}{f_2(y)} = \frac{1}{2}[(\boldsymbol{y}-\boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_2) - (\boldsymbol{y}-\boldsymbol{\mu}_1)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_1)], \tag{4}$$

经过简单计算后，有

$$W(\boldsymbol{y}) = (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\bar{\boldsymbol{\mu}}), \tag{5}$$

其中，$\bar{\boldsymbol{\mu}} = \frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)$.

由于在判别问题中，各总体的地位一般是相同的，不存在比较重视某个总体的情况，故为保证犯两类错误的概率相同，这时的判别规则可以描述为

$$\begin{cases} \boldsymbol{y}\in\boldsymbol{G}_1, & W(\boldsymbol{y}) \le 0; \\ \boldsymbol{y}\in\boldsymbol{G}_2, & W(\boldsymbol{y}) > 0. \end{cases} \tag{6}$$

另外从(5)式可以看出通过仿照似然比导出的判别函数 $W(\boldsymbol{y})$ （不考虑常数项因子）其实就是一个二次型的差，而这个二次型

$$(\boldsymbol{x}-\boldsymbol{y})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{y}) \tag{7}$$

和前面提到的欧几里得空间中的马氏距离是一致的。

2.2 Fisher 线性判别函数[1][2]

在距离判别中导出的判别函数可以是指标变量的一次函数（总体协方差矩阵相同时），也可以是指标变量的二次函数（总体协方差矩阵不同时）。在这种情况下，Fisher 提出了只考虑判别函数是线性（一次函数）的那种情况，即以各个指标变量或特征的加权和作为一个新的综合指标来作为判别归类的依据。这就要求在所有的线性判别函数中找到最优的那个线性判别函数，为了找到最优的线性判别函数，Fisher 提出了一个准则：在判别函数总方差之上衡量判别函数在各个总体中的均值应该具

有较大的离差。这个准则被称为 Fisher 准则，即

$$\Delta(a) \triangleq \frac{\sum_{i=1}^{k}\left[E_i(\boldsymbol{a}'\boldsymbol{y})-\frac{1}{k}\sum_{i=1}^{k}E_i(\boldsymbol{a}'\boldsymbol{y})\right]^2}{\sum_{i=1}^{k}D_i(\boldsymbol{a}'\boldsymbol{y})}. \tag{8}$$

通过最大化 Fisher 准则，就可以解出最优的线性判别函数式，即最优线性判别函数 $\boldsymbol{u}'\boldsymbol{y}$ 满足

$$\Delta(\boldsymbol{u}) = \max_{\boldsymbol{a}\in R^p} \Delta\boldsymbol{a}. \tag{9}$$

为了求出上式的解，可以将(9)式表示成矩阵的形式，有

$$\Delta(\boldsymbol{a}) = \frac{\boldsymbol{a}'\sum_{i=1}^{k}(\boldsymbol{\mu}_i-\bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_i-\bar{\boldsymbol{\mu}})'\boldsymbol{a}}{\boldsymbol{a}'(\sum_{i=1}^{k}\boldsymbol{\Sigma}_i)\boldsymbol{a}} \triangleq \frac{\boldsymbol{a}'\boldsymbol{M}\boldsymbol{a}}{\boldsymbol{a}'\boldsymbol{\Sigma}\boldsymbol{a}}. \tag{10}$$

由于在 Fisher 准则中 $\Delta(\boldsymbol{a})$ 的大小只取决于向量 $\boldsymbol{a}$ 的方向，而与 $\boldsymbol{a}$ 的大小或者说长度 $\|\boldsymbol{a}\|$ 无关。不妨令 $\boldsymbol{a} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}$，其中 $\boldsymbol{b}$ 为单位向量，也即 $\|\boldsymbol{b}\| = 1$。于是有

$$\Delta(\boldsymbol{a}) = \Delta\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}\right) = \boldsymbol{b}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{M}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}. \tag{11}$$

又有

$$\max_{\|\boldsymbol{b}\|=\boldsymbol{1}} \boldsymbol{b}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{M}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b} = \lambda_1(\boldsymbol{\Sigma}^{-1}\boldsymbol{M}). \tag{12}$$

从而极大值点为 $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{M}\boldsymbol{\Sigma}^{-\frac{1}{2}}$ 的特征值 λ_1 对应的特征向量 $\boldsymbol{c}_1$，因此 $\boldsymbol{a}_1 = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{c}_1$ 是 $\boldsymbol{\Sigma}^{-1}\boldsymbol{M}$ 的对应于特征值 λ_1 的特征向量。

得到最优的线性判别函数为 $W_1(\boldsymbol{y}) = \boldsymbol{a}_1'\boldsymbol{y}$，而与其对应的特征值 λ_1 称作线性判别函数 $W_1(\boldsymbol{y})$ 的判别效率，λ_1 是所有线性判别函数中判别效率的最大值。

在得到线性判别函数之后，就可以根据线性判别函数来对给定的某一样品 $\boldsymbol{y}$ 进行判别归类，但在这之前首先需要给出判别规则。根据线性判别函数 $W_1(\boldsymbol{y})$ 进行判别的规则是：分别计算 $W_1(\boldsymbol{y})$ 与 各个 $W_1(\boldsymbol{\mu}_i)$ 之间的距离

$$d\left(W_1(\boldsymbol{y}), W_1(\boldsymbol{\mu}_i)\right) = \frac{|\boldsymbol{a}_1'\boldsymbol{y}-\boldsymbol{a}_1'\boldsymbol{\mu}_i|}{(\boldsymbol{a}_1'\boldsymbol{\Sigma}_i\boldsymbol{a}_1)^{\frac{1}{2}}}, \text{ 其中 } i = 1,2,\cdots,k, \tag{13}$$

然后把 $\boldsymbol{y}$ 判给距离最小那个总体 $\boldsymbol{G}_i$。

由于各个 μ_i 不完全相同，$\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{M}\boldsymbol{\Sigma}^{-\frac{1}{2}}$ 至少有一个正的特征值。并且 $\boldsymbol{\Sigma}^{-1}\boldsymbol{M}$ 的正的特征值不止一个，不妨设有 r 个不同的正特征值，$\lambda_1 > \lambda_2 > \cdots > \lambda_r > 0$。这种情况下，只用一个线性判别函数 $W_1(\boldsymbol{y}) = \boldsymbol{a}_1'\boldsymbol{y}$ 可能不能较准确地把给定的某一样品 y 判别归类。因此可以再计算出剩下的 $r-1$ 个线性判别函数 $W_i(\boldsymbol{y}) = \boldsymbol{a}_i'\boldsymbol{y}(i = 2,3,\cdots,r)$，这时我们就可以把这 r 个线性判别函数综合起来对样品 y 判别归类。

可以记矩阵 $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_r)$，计算 $\boldsymbol{A}'\boldsymbol{y}$ 到各个总体 $\boldsymbol{G}_i$ 的距离，这就和距离判别相同了，因此采用距离判别时使用过的马氏距离

$$d^2(\boldsymbol{A}'\boldsymbol{y}, \boldsymbol{A}'\boldsymbol{\mu}_i) = (\boldsymbol{y} - \boldsymbol{\mu}_i)'\boldsymbol{A}(\boldsymbol{A}'\boldsymbol{\Sigma}_i\boldsymbol{A})^{-1}\boldsymbol{A}'(\boldsymbol{y} - \boldsymbol{\mu}_i), \tag{14}$$

判别时的依据也和距离判别如出一辙，即把某一样品 $\boldsymbol{y}$ 判给距离最近的那一个总体 $\boldsymbol{G}_i$。

从上面可以看出 Fisher 的线性判别函数就是在对原始数据做降维处理，把数据投影到一个较低维的空间中，在这个低维空间中需要尽可能地把各个总体区分开。求出的 r 个最优线性判别函数通过 $\boldsymbol{A}'\boldsymbol{y}$ 这个线性变换把原始数据从 p 维空间投影到了更低的 r 维空间，在这个 r 维空间中各个总体能够比较好地分离开，后续所有的统计推断都在这个较低维的空间中进行，判别分析也不例外。

从这也能联想到我们学过的多元分析的一些方法大都是在对原始数据做降维处理，这些方法的区别就是降维投影到的那个低维空间不完全相同。之所以不同是因为研究的对象、要解决的问题不同，如判别问题目的是为了把各个总体区分开，那么就要求投影到的那个低维空间能尽可能地把各个总体分离开来；主成分分析则是由于所要研究的问题的指标变量太多，指标变量过多自然就导致研究时变得麻烦，所以能不能删除一些对我们研究问题没有什么意义的指标变量，或者说我们能不能把这些指标综合成几个更能说明问题，对我们更有价值的指标变量，同时又不至于损失太多信息，这就有了主成分分析这一工具，主成分分析就要求降维后的那个低维空间能够

尽可能地保留原始数据样本的所有信息，用统计的话讲就是样本点应在这个低维空间中具有较大的方差。

2.3 Bayes 判别规则[1][2]

Bayes 判别的思想也是由 Fisher 首先提出来的，具体就是将 Bayes 定理直接应用到判别问题上。从 Bayes 判别中也能看出来 Wald 在 1950 年提出的统计判决理论的影子，其中的一些概念在判决理论中也有了更一般的表达。

记 $\pi_j \triangleq P(\theta = j)(j = 1,2,\cdots,k)$ 为样品 $\boldsymbol{y}$ 来自于总体 $\boldsymbol{G_j}$ 的先验概率，$L(i,j)$ 为样品 $\boldsymbol{y}$ 来自于总体 $\boldsymbol{G_j}$ 却被误判为 $\boldsymbol{G_i}$ 的损失，误判概率为

$$p_{ij} = \int_{D_i} f_j(y)d\boldsymbol{y}. \tag{15}$$

将误判概率与误判损失相结合得风险函数（即平均损失函数）为

$$\sum_{i=1}^{k} L(i,j)p_{ij} = \sum_{i=1}^{k} L(i,j)\int_{D_i} f_j(y)d\boldsymbol{y}. \tag{16}$$

又因为样品 $\boldsymbol{y}$ 来自于总体 $\boldsymbol{G_j}$ 的先验概率为 $\pi_j \triangleq P(\theta = j)(j = 1,2,\cdots,k)$，得误判的 Bayes 风险为

$$R_\pi(D) = \sum_{j=1}^{k} \pi_j \sum_{i=1}^{k} L(i,j)p_{ij} = \sum_{j=1}^{k} \pi_j \sum_{i=1}^{k} L(i,j)\int_{D_i} f_j(y)d\boldsymbol{y}. \tag{17}$$

最小化 Bayes 风险就可以得到 Bayes 判别规则，且当 $\boldsymbol{G_j}$ 为正态同方差时，是样品 $\boldsymbol{y}$ 的一个线性函数。

3　线性判别分析在英雄联盟单双排对局预测中的应用

3.1　《英雄联盟》介绍

3.1.1　游戏介绍

《英雄联盟》（League of Legends，简称 LOL）是一款是由拳头游戏于 2009 年发行的 5v5 多人在线竞技类游戏，该游戏是受到《魔兽争霸 III：冰封王座》中一个名为 DotA 的第三方自定义地图启发而诞生[13]，直至今天仍然是一款非常流行的游戏。《英雄联盟》于 2018 年 5 月成为亚运会的表演项目之一，2021 年 11 月成为第 19 届杭州亚运会的正赛项目之一，所获得的奖牌计入国家奖牌榜[14]。

在游戏中，两支分别由 5 个召唤师组成的队伍称为蓝色方和紫色方进行一场大约 15—40 分钟的战斗，最终目标是摧毁对方队伍的基地。在召唤师峡谷经典模式中每个召唤师都会控制一个独一无二的英雄，截止到当前版本（Ver 12.9.1），《英雄联盟》已经拥有了 159 个不同的英雄，这意味着每个召唤师都可以从这约 160 个不同的英雄中选择一个来控制。

3.1.2　召唤师峡谷地图[13]

如**图 2** 所示，在召唤师峡谷地图中有三条路线，分别称为上路、中路和下路，在这三条路线上蓝色方和紫色方分别拥有 3 座一塔，3 座二塔，3 座高地防御塔，每个队伍的目标都是摧毁对方的防御塔，同时保护自己的防御塔不被对方摧毁。

除了线上之外，召唤师峡谷地图上剩下的就是野区的中立资源了。有 5 种中立资源，分别是野怪、小龙、远古龙、峡谷先锋以及大龙。召唤师获得这些中立资源会得到金币的奖励和英雄属性加成，所以这也是蓝色方和紫色方队伍成员争夺的目标。

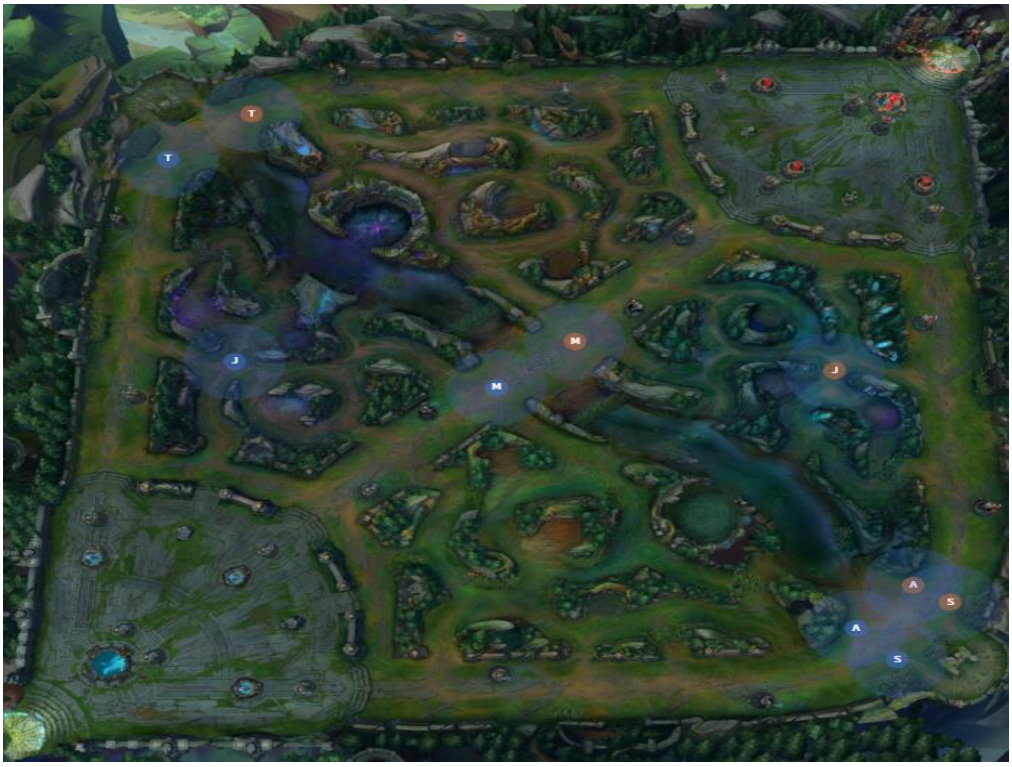

图 2 召唤师峡谷地图

资料来源：https://map.riftkit.net/[15]

3.1.3 英雄类型及分线[13]

在 3.1.1 部分中已经提到截止到当前版本（Ver 12.9.1），《英雄联盟》已经拥有了 159 个不同的英雄。这些英雄被分为 6 大类，分别是坦克、战士、刺客、法师、射手以及辅助。在 3.1.2 部分中提到了**图 2** 召唤师峡谷地图中有 3 条路线和野区，这 6 类英雄根据其英雄特性会被分配到不同的位置。其中坦克、战士和部分法师类英雄通常走上路，战士、刺客和部分法师类英雄通常会打野，刺客和法师类英雄通常走中路，射手和辅助则会走下路，这是召唤师峡谷地图的标准分线。其中有部分英雄由于其独特的英雄特性或者是版本变化的原因可以走不同的位置，也就是我们通常所说的“英雄摇摆”。这种英雄摇摆的情况在各个位置都出现过，但出现这种摇摆情况最多的位置是上路、打野以及中路。下路由于其是射手和辅助的双人路，在位置分配方面受到版本变化和影响比较小。

3.2 游戏对局数据说明

本文分析的数据通过拳头游戏开发者账号的 API：RGAPI-1e2ece0e-85d4-4644-a358-90d7c375d088（已过期）并借助于 Python 软件包 Riot-watcher（https://github.com/pseudonym117/Riot-Watcher）[16]及 Cassiopeia（https://github.com/meraki-analytics/cassiopeia）[17]获得，所得数据已上传至 https://raw.githubusercontent.com/loreliu/bachelor-thesis/main/timeline_840.csv。

英雄联盟召唤师峡谷经典模式对局的时间线数据是游戏进程中每分钟数据的快照，其中包括两种类型的数据：参与者数据和事件数据。参与者数据每分钟都会提供一个快照，包括英雄的属性数据，如生命值、法力值、攻击力、护甲、魔抗、技能急速等。以及当前时刻的总金币、输出及承受伤害、小兵击杀数、野怪击杀数、英雄等级、在地图上的位置、经验值。事件数据就是游戏进程中发生特定事件的记录，如物品购买、英雄升级、英雄技能升级、击杀或参与击杀敌方英雄、摧毁敌方防御塔、夺取中立资源如小龙、峡谷先锋、大龙、远古龙等。

在当前版本的英雄联盟单双排对局中，前期（对线期）的经济来源主要有三种，英雄击杀经济（基础 300）和防御塔镀层经济（每层 160，共 5 层）、夺取中立资源（如小龙、峡谷先锋）经济。其中三路防御塔自带的镀层经济是最多的，是双方在前期争夺的主要目标，由此导致对线 1v1、2v2 及小规模资源团（如 8 分钟的峡谷先锋团）的竞争非常频繁。对线击杀通常情况下都至少能获得一层镀层经济，而峡谷先锋则至少能获得两层镀层经济，最多时能获得满层（5 层）镀层经济，一座一塔，一座二塔，而边线（上路和下路）的二塔还有额外 250 经济。在 14 分钟时，三座一塔的镀层自动脱落，这时通常就会进入换线期，围绕第二条峡谷先锋，及中路防御塔展开争夺。因此我们选择 14 分钟时的时间线数据对对局胜负进行预测。

在较高端对局（铂金钻石分段以上）中，对局双方都是“本地人”的情况下，由于双方都比较会玩，知道对方要做什么也知道如何应对，14 分钟前双方的经济拉不

开较大差距，往往都是在中期处理上拉开经济差。在较低端对局（白银及以下），对局双方同样都是“本地人”的情况下，由于双方都不怎么会玩，只看重英雄击杀，不注重峡谷先锋和小龙的争夺导致 14 分钟前经济也拉不开较大差距。而在黄金分段双方都是“本地人”的情况下，对局一方知道要争夺峡谷先锋和小龙，而另一方却没有意识去阻止。获得峡谷先锋的一方便可以利用峡谷先锋获得大量的镀层经济，拉开经济差。

因此本文选取召唤师峡谷经典模式单双排黄金分段对局时间线数据中 14 分钟时小龙夺取数、峡谷先锋夺取数、防御塔夺取数、双方不同位置的经济差数据，以及游戏时长、获胜队伍和英雄选择。时间线数据变量解释说明如**表 1** 所示。

表 1 时间线数据变量说明

变量名称	变量说明	变量取值	备注
t1Dragons	蓝色方小龙数	0,1,2	
t2Dragons	紫色方小龙数	0,1,2	
t1Rift	蓝色方峡谷先锋数	0,1	
t2Rift	紫色方峡谷先锋数	0,1	
t1TopTowerTaken	蓝色方上塔丢失数	0,1,2,3,4	三防御塔，一水晶
t1MidTowerTaken	蓝色方中塔丢失数	0,1,2,3,4,5,6	三防御塔，一水晶，两基地防御塔
t1BotTowerTaken	蓝色方下塔丢失数	0,1,2,3,4	
t2TopTowerTaken	紫色方上塔丢失数	0,1,2,3,4	
t2MidTowerTaken	紫色方中塔丢失数	0,1,2,3,4,5,6	
t2BotTowerTaken	紫色方下塔丢失数	0,1,2,3,4	
topGoldDiff	上单对位经济差		蓝色方与紫色方对位经济差值
jugGoldDiff	打野对位经济差		
midGoldDiff	中单对位经济差		
adcGoldDiff	ADC 对位经济差		
supGoldDiff	辅助对位经济差		
gameLength	游戏时长		单位：秒
winningTeam	获胜队伍	100,200	100：蓝色方，200：紫色方
t1Top	蓝色方上单英雄		
t1Jug	蓝色方打野英雄		

t1Mid	蓝色方中单英雄
t1Adc	蓝色方 ADC 英雄
t1Sup	蓝色方辅助英雄
t2Top	紫色方上单英雄
t2Jug	紫色方打野英雄
t2Mid	紫色方中单英雄
t2Adc	紫色方 ADC 英雄
t2Sup	紫色方辅助英雄

3.3 探索性数据分析

3.3.1 按获胜队伍分组后各变量的统计特征

计算按获胜队伍分组后各变量的均值及标准差如**表 2** 和**表 3** 所示，

表 2 蓝色方获胜时各变量的均值及标准差

变量名称	均值	标准差
t1Dragons	0.9373	0.6994
t2Dragons	0.5430	0.6422
t1Rift	0.4886	0.4999
t2Rift	0.3067	0.4612
t1TopTowerTaken	0.1225	0.3609
t1MidTowerTaken	0.05949	0.2582
t1BotTowerTaken	0.07139	0.2631
t2TopTowerTaken	0.3108	0.5515
t2MidTowerTaken	0.2018	0.4669
t2BotTowerTaken	0.2142	0.4395
topGoldDiff	504.9	1602
jugGoldDiff	397.0	1247
midGoldDiff	443.2	1332
adcGoldDiff	529.4	1457
supGoldDiff	314.5	1023

表 3 紫色方获胜时各变量的均值及标准差

变量名称	均值	标准差
t1Dragons	0.5164	0.6994

t2Dragons	0.9754	0.6422
t1Rift	0.3524	0.4999
t2Rift	0.4461	0.4612
t1TopTowerTaken	0.3047	0.3609
t1MidTowerTaken	0.2043	0.2582
t1BotTowerTaken	0.2152	0.2631
t2TopTowerTaken	0.1502	0.5515
t2MidTowerTaken	0.06623	0.4669
t2BotTowerTaken	0.06775	0.4395
topGoldDiff	-458.7	1602
jugGoldDiff	-474.2	1247
midGoldDiff	-476.2	1332
adcGoldDiff	-545.6	1457
supGoldDiff	-309.4	1023

可以看到，在蓝色方和紫色方在获胜时小龙夺取数、峡谷先锋夺取数、防御塔丢失数、各位置对位经济差的均值均有比较大的差异。蓝色方和紫色方在获胜时各位置的对位经济差的标准差几乎相同，而小龙夺取数、峡谷先锋夺取数和防御塔丢失数的标准差则有一些差异。但蓝色方获胜时己方的小龙夺取数与紫色方获胜时敌方的小龙夺取数的均值及标准差相近，在峡谷先锋夺取数和防御塔丢失数上也有同样的表现，这说明蓝色方和紫色方无论哪一方获胜时需要取得的优势是相同的，即当前版本蓝色方和紫色方的地位是平等的，没有哪一方有天然的优势。为了更清楚的说明这一点，计算双方获胜时双方小龙、峡谷先锋、防御塔丢失数之差的均值及标准差如**表 4** 和**表 5** 所示。

表 4 蓝色方获胜时双方小龙、峡谷先锋、防御塔丢失数之差的均值及标准差

变量名称	均值	标准差
dragonDiff	0.3944	1.232
riftDiff	0.1819	0.8732
topTowerTakenDiff	-0.1853	0.7049
midTowerTakenDiff	-0.1423	0.5507
botTowerTakenDiff	-0.1428	0.5358

表 5 紫色方获胜时双方小龙、峡谷先锋、防御塔丢失数之差的均值及标准差

变量名称	均值	标准差
dragonDiff	-0.4590	1.232
riftDiff	-0.09363	0.8832
topTowerTakenDiff	0.1545	0.7049
midTowerTakenDiff	0.1380	0.5507
botTowerTakenDiff	0.1474	0.5358

图 *3* 是绘制出的各变量的相关系数矩阵，从中可以看到蓝色方和紫色方的小龙夺取数、峡谷先锋夺取数有比较强的相关性，呈负相关，而两队各位置的防御塔丢失数则没有这么明显的相关性。这是因为小龙和峡谷先锋是中立资源，一方夺取了，另一方自然就不能再夺取，而防御塔则是双方都具有的战略点。

还可以看到 ADC 和辅助的对位经济差也有比较强的相关性，这是因为两者都走下路，一个人有经济优势另一个自然也会有优势。

众所周知，在召唤师峡谷中上路是对抗路，下路是发育路，而中路由于其处于整张地图的中心位置，容易受到各个位置的影响，比如辅助游走、打野 gank。这可以从相关系数看出，比如上单对位经济差与摧毁上路防御塔的相关系数为 0.5572，ADC 对位经济差与摧毁下路防御塔的相关系数为 0.4517，而中单对位经济差与摧毁中路防御塔的相关系数仅为 0.3702。

除了上述较强的相关性之外，**图 *3*** 各变量的相关系数矩阵中也有较低的相关系数。下路 ADC 和辅助的对位经济差与峡谷先锋夺取相关系数的最大绝对值为 0.094，上单对位经济差与小龙夺取的相关系的最大绝对值为 0.0826，这从**图 2** 召唤师峡谷地图中上路和小龙的距离以及下路和峡谷先锋的距离可以看出。上路取得对位经济优势对争夺小龙起的作用比较小，下路取得经济优势与抢到峡谷先锋同样关系不大。不过在高端对局及职业比赛中，下路取得经济优势通常会在 8 分钟前回城购买装备，将经济优势转化为装备优势，然后赶到峡谷先锋处。由于其一方具有装备和等级优势，比较容易在峡谷先锋团中取胜，进而夺得峡谷先锋。

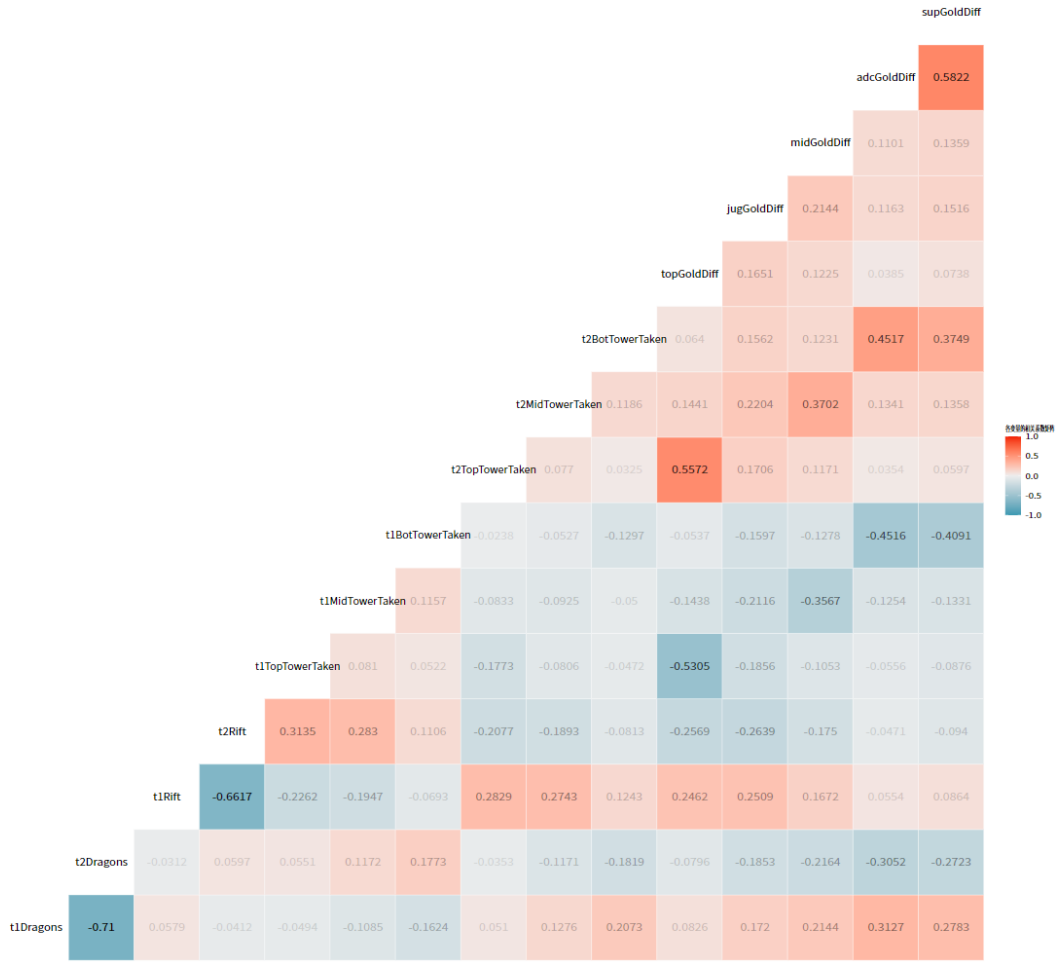

图 **3** 各变量的相关系数矩阵

3.3.2　蓝色方与紫色方获胜胜率分析

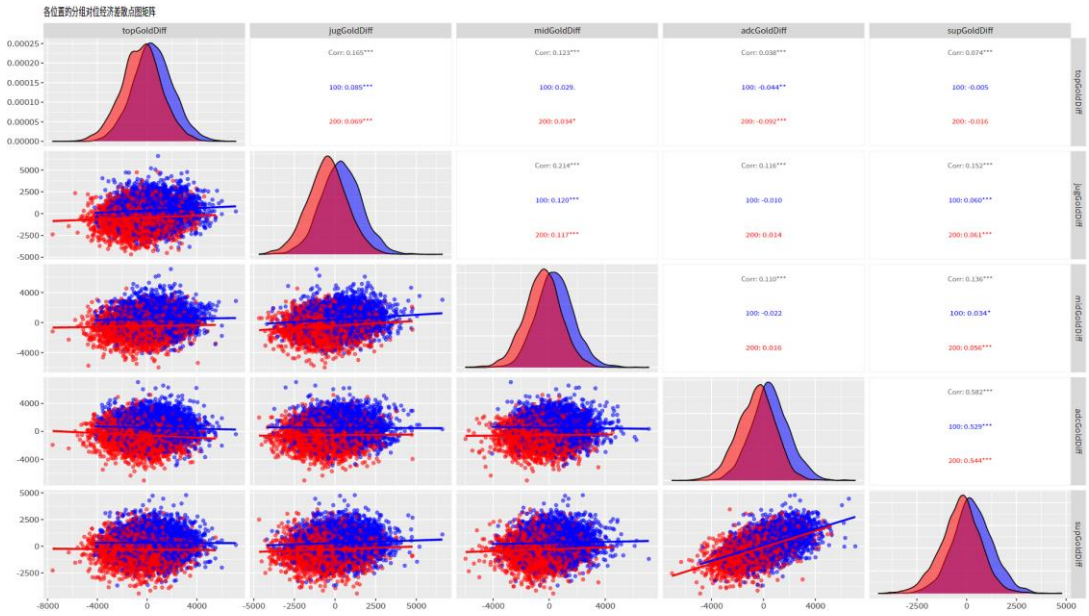

图 **4** 各位置的分组对位经济差散点图矩阵

图 4 是绘制的各位置的分组对位经济差散点图矩阵，矩阵下半部分是分组对位经济差散点图，对角线是分组对位经济差的分布图，上半部分则是相关系数。从下半部分的散点图中也可以看出 ADC 和辅助的对位经济差有较强的相关性，散点大致分布在 45°直线两侧。从对角线上的分布图可以看出来还是有不少的翻盘局发生，即 14 分钟前对位经济领先但最终输掉游戏的情况。这可能是前期经济领先的一方没有控到小龙，或是前期经济落后的一方是后期阵容等等，有很多不确定的因素都可能导致被翻盘。具体可以计算前期经济领先一方被翻盘的概率或其胜率，如**表 6** 所示。

表 614 分钟时蓝色方各位置对位经济领先的胜率

位置 \ 对位经济领先	0-500	500-1000	1000-1500	1500-2000	2000-2500
上单	0.5132	0.5684	0.6506	0.6558	0.6947
打野	0.5574	0.6195	0.7166	0.7373	0.7542
中单	0.5238	0.6221	0.6623	0.7500	0.7444
ADC	0.5572	0.6070	0.6344	0.6990	0.7969
辅助	0.5531	0.6091	0.7027	0.7535	0.8274

表 6 续 14 分钟时蓝色方各位置对位经济领先的胜率

位置 \ 对位经济领先	2500-3000	3000-3500	3500-4000	4000-4500	>4500
上单	0.7289	0.7740	0.7647	0.8571	0.9286
打野	0.8205	0.9038	0.9091	1.000	0.8182
中单	0.8011	0.8750	0.9333	1.000	0.9500
ADC	0.8105	0.8880	0.9219	0.8621	0.8696
辅助	0.7969	0.7586	1.000	1.000	1.000

图 5 是**表 6**“14 分钟时蓝色方各位置对位经济领先的胜率”的折线图，从图中可以很轻易地看出随着经济优势的扩大，胜率也在增加。而在同样的经济优势下，上单的胜率要低于其他四个位置。上单的胜率在对位经济差达到 3000-3500 左右时出现了短暂的下降，随后稳步上升。当辅助经济领先达到 2500 左右时，胜率开始下降，然而在达到 2500 左右时，胜率又上升至 100%。ADC 对位经济差在达到 3500-4000 左右时胜率开始下降，4000 以上时开始回升。中单和打野在经济领先达到 4000 后，

胜率开始下降。

上述的各位置胜率在达到不同程度的经济领先时都存在短暂的下降，这可能是阵容选择的原因，也可能是玩家在取得较大的经济优势后不谨慎，没有继续扩大优势，导致被翻盘。

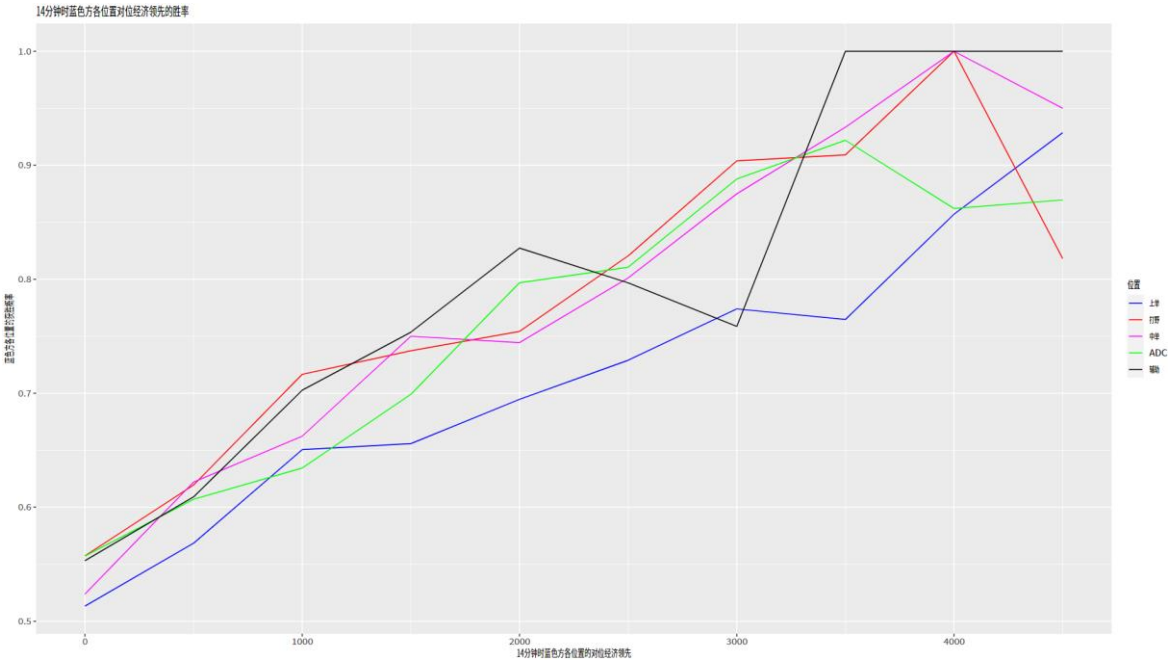

图 514 分钟时蓝色方各位置对位经济领先的胜率折线图

下面计算蓝色方夺得小龙时的胜率，计算结果如**表 7** 所示。

表 714 分钟时蓝色方夺得小龙的胜率

双方小龙数	紫色方=0	紫色方=1	紫色方=2
蓝色方=0	0.5103	0.3845	0.2655
蓝色方=1	0.6465	0.5080	
蓝色方=2	0.7651		

表 814 分钟时蓝色方夺得峡谷先锋的胜率

双方峡谷先锋数	紫色方=0	紫色方=1
蓝色方=0	0.5150	0.4181
蓝色方=1	0.5916	

表 8 是 14 分钟时蓝色方夺得峡谷先锋的胜率。从**表 7** 及**表 8** 可以发现，夺取小龙的胜率增加百分比要比夺取峡谷先锋的胜率高。

表 914 分钟时蓝色方摧毁上路防御塔的胜率

摧毁上路防御塔数	紫色方=0	紫色方=1	紫色方=2	紫色方=3	紫色方=4
蓝色方=0	0.5177	0.3132	0.2368	0	0.3333
蓝色方=1	0.6778	0.4615	0		
蓝色方=2	0.7312	0.3333			
蓝色方=3	0.8000				
蓝色方=4	1.000				

表 1014 分钟时蓝色方摧毁中路防御塔的胜率

摧毁中路防御塔数	紫色方=0	紫色方=1	紫色方=2	紫色方=3	紫色方=4
蓝色方=0	0.5114	0.2539	0.1743	0.1667	0
蓝色方=1	0.7533	0.5294	0	0	
蓝色方=2	0.8081	1.000			
蓝色方=3	0.8000	0			
蓝色方=4	1.000				
蓝色方=5					
蓝色方=6					

表 10 续 14 分钟时蓝色方摧毁中路防御塔的胜率

摧毁中路防御塔数	紫色方=5	紫色方=6
蓝色方=0		0
蓝色方=1		
蓝色方=2		
蓝色方=3		
蓝色方=4		
蓝色方=5		
蓝色方=6		

表 1114 分钟时蓝色方摧毁下路防御塔的胜率

摧毁下路防御塔数	紫色方=0	紫色方=1	紫色方=2	紫色方=3	紫色方=4
蓝色方=0	0.5111	0.2632	0.1364	0	
蓝色方=1	0.7522	0.5455			
蓝色方=2	0.9623				
蓝色方=3					
蓝色方=4					

表 9、表 10、表 11 是 14 分钟时蓝色方摧毁三路防御塔数的胜率，可以发现当领先同样的防御塔数时，中路和下路的胜率要比上路高。在**表 6**“14 分钟时蓝色方各位置对位经济领先的胜率”中也得到了相似的结论“在同样的经济优势下，上单的胜率要低于其他四个位置”。这反映到游戏中就是上路的 Carry 能力不如中路和下路，实际游戏体验也是如此。

3.4 将线性判别分析应用到时间线数据预测对局胜负

3.4.1 变量选择及模型假设验证

在 R 中实现线性判别分析可以使用 MASS 软件包里的 lda 函数，但在这之前还需要对线性判别分析的一些假设进行验证。由于 lda 使用 Bayes 判别计算判别函数，所以需要检查不同组别中各预测变量的正态性、组别间的方差同质性以及各变量间是否有多重共线性。

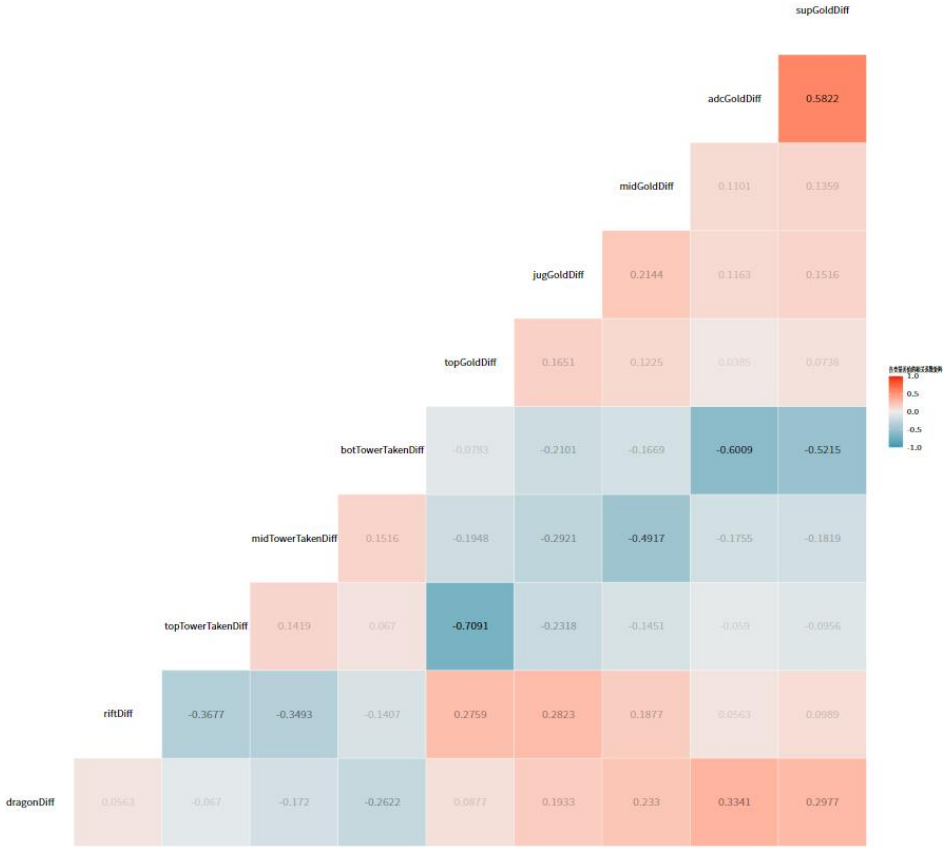

图 6 各变量差值的相关系数矩阵

观察**图 3** 和**图 6** 的相关系数矩阵图发现相关系数绝对值最大的两个仅为 0.71 和 0.7091。因此在保留样本信息，使更多的变量进入模型的前提下，选择 dragonDiff、riftDiff、t1TopTowerTaken、t2TopTowerTaken、t1MidTowerTaken、t2MidTowerTaken、t1BotTowerTaken 、 t2BotTowerTaken 、 topGoldDiff 、 jugGoldDiff 、 midGoldDiff 、adcGoldDiff、supGoldDiff 进入模型。

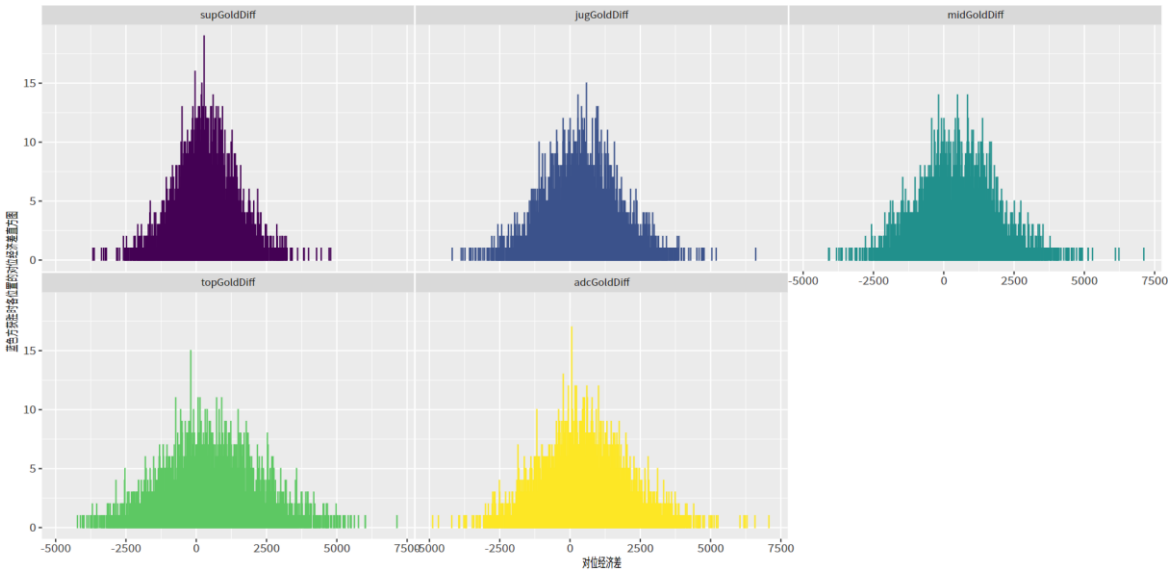

图 7 蓝色方获胜时各位置的对位经济差直方图

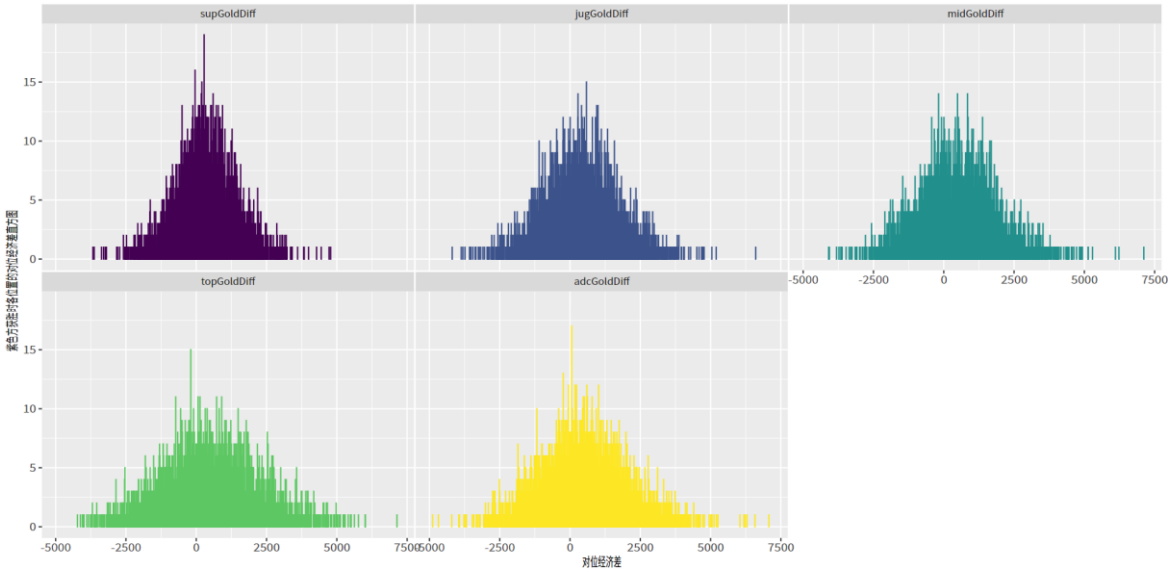

图 8 紫色方获胜时各位置的对位经济差直方图

在检验各位置对位经济差是否服从正态分布时，由于样本量太大，Shapiro-wilk

检验显著，拒绝零假设“总体服从正态分布”。但从**图 7** 和**图 8** 可以看出各位置对位经济差是近似服从正态分布的。其他几个变量因为取值不是连续的，不服从正态分布。

小龙夺取数之差、峡谷先锋夺取数之差及各位置对位经济差的 BoxM 检验不显著，p 值为 0.4892，认为满足方差同质性。其他 6 个防御塔丢失数变量不满足方差同质性。

3.4.2 Bayes 判别函数及判别规则

虽说有些变量不符合 Bayes 判别的假设，但无论如何，我们可以尝试将其应用到 14 分钟时的时间线数据上，看看预测结果怎么样。为了使结果可复现，从 random.org 获取一个随机数作为种子（4225），将 14 分钟时的时间线数据分为训练集和测试集，比例为 8:2。

将 Bayes 判别 MASS::lda 应用到训练集可以得到

蓝色方获胜时的均值为

$$\widehat{\mu_{100}} = (0.4054, 0.1741, 0.1265, 0.3064, 0.06122, 0.2028, 0.2109, 501.2, 412.1, 438.0, 523.7, 316.8). \tag{18}$$

紫色方获胜时的均值为

$$\widehat{\mu_{200}} = (-0.4499, -1011, 0.3031, 0.1458, 0.2048, 0.06975, 0.2124, 0.07229, -469.0, -481.8, -474.4, -529.9, -299.6). \tag{19}$$

记

$$\hat{h} = \frac{1}{2}(\widehat{\mu_{100}} + \widehat{\mu_{200}}). \tag{20}$$

Bayes 标准化判别函数为

$$y = -0.2261 * dragonDiff + 0.008440 * riftDiff - 0.05458 * t1TopTower$$

$$Taken + 0.1875 * t2TopTowerTaken - 0.1160 * t1MidTowerTaken + 0.2517 * t2MidTowerTaken - 0.1367 * t1BotTowerTaken + 0.1706 * t2BotTowerTaken - 0.0003078 * topGoldDiff - 0.0003615 * jugGoldDiff - 0.0003349 * midGoldDiff - 0.0003208 * adcGoldDiff - 0.0001308 * supGoldDiff. \quad (21)$$

这同样也是 Fisher 线性判别函数。若记原始变量为 $\boldsymbol{y}$,则 Bayes（Fisher）标准化判别函数中的各变量值为 $\boldsymbol{y} - \hat{h}$.

最终得到 Bayes（Fisher）判别规则

$$\begin{cases} \text{蓝色方获胜}, & y \leq 0; \\ \text{紫色方获胜}, & y > 0. \end{cases} \quad (22)$$

表 12 Bayes 判别在训练集上的混淆矩阵

训练 \ 预测	蓝色方	紫色方
蓝色方	2447	787
紫色方	760	2394

表 13 Bayes 判别在测试集上的混淆矩阵

测试 \ 预测	蓝色方	紫色方
蓝色方	652	232
紫色方	194	593

表 12 是 Bayes 判别在训练集上的混淆矩阵，预测正确率为 75.78%。**表 13** 是 Bayes 判别在测试集上的混淆矩阵，预测正确率为 74.51%。

默认情况下，MASS::lda 函数在根据后验概率判别时选用的临界值为 0.5（随机猜测），因此可以尝试寻找使得预测正确率最大的临界值。在训练集上，当临界值为 0.49 时，预测正确率最大，为 76.00%；在测试集上，当临界值为 0.42 时，预测正确率最大，为 75.28%。因此为了更准确地预测对局胜负，选用 0.42 作为后验概率的临界值。

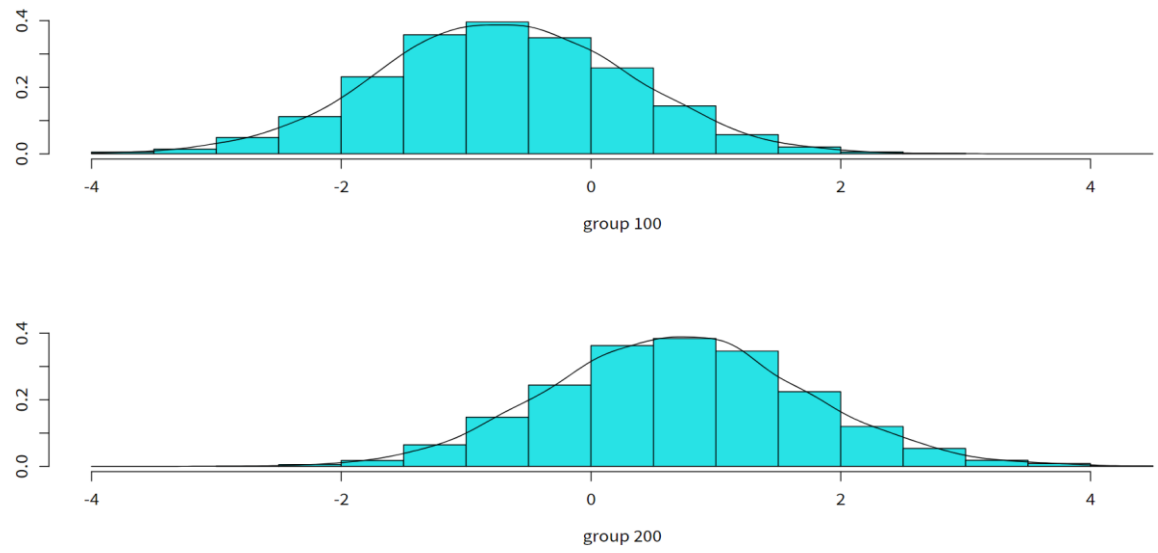

图 **9**Bayes（Fisher）判别函数的直方图

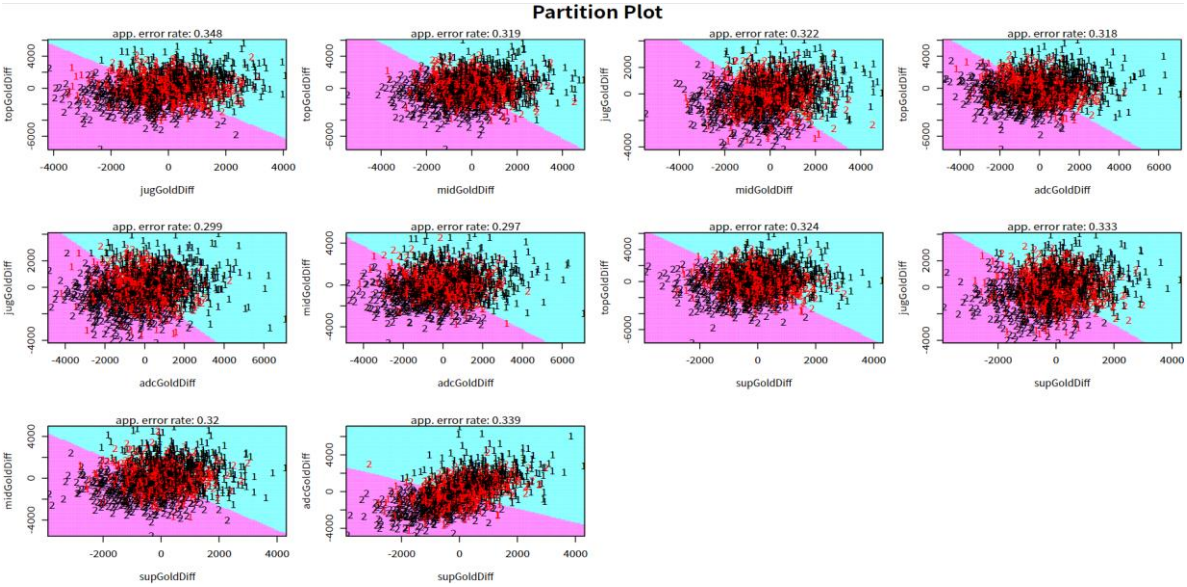

图 **10**Bayes（Fisher）判别函数在测试集对位经济差上的分区图

从**图 9**、**图 10** 中可以看到 Bayes 判别的结果有部分重叠，即 Bayes 判别函数的结果在 $y = 0$ 附近时没有预测正确，但总体上 Bayes 判别函数把大部分样品都正确地分离开了。

表 14 测试集部分正判样品

变量名称	样品 1	样品 2	样品 3	样品 4	样品 5	样品 6
t1Dragons	2	0	2	1	1	2
t2Dragons	0	2	0	0	1	0

t1Rift	0	0	1	0	1	1
t2Rift	0	1	0	0	0	0
t1TopTowerTaken	0	1	0	0	0	0
t1MidTowerTaken	0	2	0	0	0	0
t1BotTowerTaken	0	2	0	0	1	0
t2TopTowerTaken	1	0	0	1	0	0
t2MidTowerTaken	0	0	0	1	0	0
t2BotTowerTaken	0	0	0	0	0	0
topGoldDiff	2336	-846	-150	1304	548	-2056
jugGoldDiff	-1008	-1472	205	-955	-1327	1247
midGoldDiff	1383	-5485	-1980	4429	131	825
adcGoldDiff	731	-2755	1463	-1363	-2556	-128
supGoldDiff	683	-339	1159	-75	-358	-1118
gameLength	1752	911	1441	2067	1526	2226
winningTeam	100	200	100	100	200	100
t1Top	普朗克	菲奥娜	厄加特	约里克	辛吉德	厄加特
t1Jug	扎克	雷恩加尔	希瓦娜	莉莉娅	卡密尔	凯隐
t1Mid	劫	维克托	弗拉基米尔	永恩	艾瑞莉娅	丽桑卓
t1Adc	艾希	卡莎	泽丽	图奇	厄运小姐	伊泽瑞尔
t1Sup	蕾欧娜	加里奥	诺提勒斯	莫甘娜	巴德	锤石
t2Top	贾克斯	提莫	德莱厄斯	菲奥娜	德莱厄斯	莫德凯撒
t2Jug	卡兹克	戴安娜	李青	戴安娜	李青	特朗德尔
t2Mid	艾克	格雷福斯	阿卡丽	塞拉斯	弗拉基米尔	阿狸
t2Adc	卢锡安	泽丽	烬	希维尔	崔丝塔娜	烬
t2Sup	诺提勒斯	派克	锤石	索拉卡	娑娜	拉克丝

表 15 测试集部分误判样品

变量名称	样品 1	样品 2	样品 3	样品 4	样品 5	样品 6
t1Dragons	1	1	0	0	0	0
t2Dragons	0	1	1	1	2	1
t1Rift	1	0	1	1	1	0
t2Rift	0	1	0	0	0	1
t1TopTowerTaken	0	0	0	0	0	0
t1MidTowerTaken	0	0	0	1	0	0
t1BotTowerTaken	0	2	1	0	0	0
t2TopTowerTaken	1	0	0	0	1	0

t2MidTowerTaken	0	0	0	0	0	0
t2BotTowerTaken	0	0	0	1	0	0
topGoldDiff	1505	433	-200	-2618	2925	-1405
jugGoldDiff	190	-947	868	434	1392	1369
midGoldDiff	-307	-2232	1754	-361	-491	-1683
adcGoldDiff	-323	1429	-2751	2114	-178	-372
supGoldDiff	-340	-1963	-1999	-63	-716	-1171
gameLength	1887	1899	1526	2087	2344	1843
winningTeam	200	100	100	100	200	100
t1Top	莫德凯撒	德莱厄斯	格雷福斯	凯尔	瑟提	普朗克
t1Jug	赫卡里姆	李青	卡兹克	伊芙琳	费德提克	莉莉娅
t1Mid	卡西奥佩娅	阿兹尔	艾瑞莉娅	卡特琳	薇古丝	永恩
t1Adc	卢锡安	薇恩	金克斯	薇恩	金克斯	薇恩
t1Sup	悠米	诺提勒斯	露露	潘森	布里茨	诺提勒斯
t2Top	希瓦娜	内瑟斯	杰斯	普朗克	奥恩	瑟提
t2Jug	波比	佛耶戈	凯隐	莉莉娅	佛耶戈	扎克
t2Mid	派克	卡特琳娜	乐芙兰	妮蔻	黑默丁格	凯尔
t2Adc	莎弥拉	卢锡安	伊泽瑞尔	烬	吉格斯	艾希
t2Sup	诺提勒斯	塞纳	萨勒芬妮	塞纳	拉克丝	布里茨

对比**表 14** 和**表 15** 的测试集部分样品可以发现 Bayes 判别正确地预测胜负时，获胜的一方在 14 分钟时要比失败的一方领先很多。这些领先包括小龙、峡谷先锋、防御塔以及队伍总经济。在中立资源及防御塔上。获胜方几乎控到了所有资源，而失败方则几乎颗粒无收。在对位经济差上获胜方队伍里虽然有一个或两个位置落后，但其他位置领先要的更多，这使得队伍总经济要领先对方。

在 Bayes 判别错误地预测胜负的对局里，14 分钟时双方队伍总经济接近于持平，小龙、峡谷先锋、防御塔也互有领先。并且误判的对局场均时长要高于正判的场均时长，这可以从**表 16** 和**图 11** 看出。14 分钟时双方均势对局，前期英雄没有取得比较大的优势，发育到中后期自然不如后期英雄对游戏的影响大。在 3.3.2 部分“蓝色方与紫色方获胜胜率分析”的结尾提到了上路的 Carry 能力不如中路和下路，如样品 4 中紫色方上单普朗克取得了 2618 的对位经济差，但不如蓝色方 ADC 薇恩的 2144 经济

差；样品 1 中蓝色方上单莫德凯撒领先了 1505 的对位经济差，不如紫色方中路派克 307 及 ADC 莎弥拉 323 的经济差；样品 5 中蓝色方上野各取得了 2925 和 1392 的对位经济差，但不如紫色方中下黑默丁格和吉格斯的 491 和 178 经济差。

表 16 测试集预测正确和预测错误时游戏时长的统计特征

游戏时长 / 预测结果	最小值	0.25 分位数	中位数	均值	0.75 分位数	最大值
正确	896	1411	1660	1680	1912	3161
错误	1233	1759	1942	1974	2190	3042

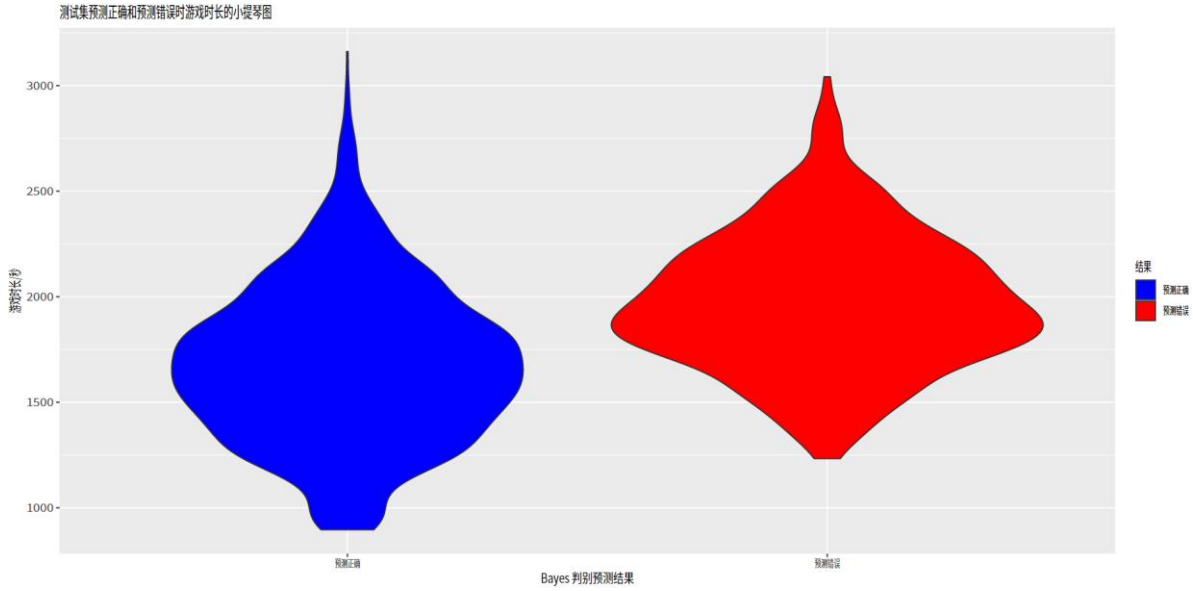

图 11 测试集预测正确和预测错误时游戏时长的小提琴图

除了个别位置的英雄选择外，整个队伍的阵容搭配对游戏能否取胜也至关重要。如样品 2 中蓝色方除了 ADC 薇恩和上路德莱厄斯之外，其他 3 个位置的经济都有很大的落后，在中立资源夺取上也落后于紫色方，但蓝色方阵容打团就是比紫色方厉害，慢慢发育打团不出意外稳赢，可以看到这场对局也是到了 32 分钟左右才由蓝色方获胜结束。

英雄联盟这个游戏，不只有对线，还有打团。前期对线拿到优势并不等于这把游戏就拿下了，一个好的阵容选择是赢下游戏的第一步。

4 小结与展望

线性判别分析作为一种分类和降维特征选择方法，在 1936 年 Fisher 提出之时，只是为了计算三种鸢尾花之间亲缘关系的远近，并不是为了将它们分类。而在我们的教学过程中，无论是书本还是老师，都告诉我们“Fisher 提出线性判别分析是为了将三种鸢尾花区分开”，但事实上并不是这样。这个几乎“公认”的说法在我们的教学过程中如此之流行应该被反思。

作为一种降维特征选择方法，线性判别分析和主成分分析经常同时出现或同时被提起，这两种方法的区别在于降维到的低维空间不同。同时，这也是许多多元分析方法的区别。多元分析自 30 年代初发展到现在，其推动力固然是实际需要，但终究离不开计算机硬件的飞速发展。计算机提升了统计分析效率的同时也带来了更多的数据，造成的结果就是统计分析的样本量在增加，样本特征也在增加。这不是绝对的好事，也不是绝对的坏事。样本多了后就更加依赖计算机算法、硬件，可以说统计分析和计算机是相辅相成的。数据增加的同时也意味着我们统计有更多的用武之地，如本文介绍的将线性判别分析应用到英雄联盟对局数据上来预测胜负。

在对英雄联盟对局时间线数据的探索性分析中，得到了一些在平时游戏中可以体会到的结论。如通过对蓝色方和紫色方获胜时各变量的均值及标准差的分析得出当前版本蓝色方和紫色方的地位是平等的，没有哪一方有天然的优势；通过观察各变量的相关系数矩阵，发现下路对位经济差和小龙夺取有较强的相关性，上单对位经济差和峡谷先锋夺取有较强的相关性；ADC 和辅助的对位经济差有较强的相关性；上单经济差和上路防御塔摧毁的相关性要高于下路和中但，其中中单经济差和中路防御塔摧毁的相关系数最低。通过对蓝色方与紫色方获胜胜率的分析，得出了控小龙比控峡谷先锋更好赢以及上单的 Carry 能力不如中单和 ADC。这些结论在我们平时游戏里都或多或少能感受得到。

在对时间线数据建立线性判别分析预测对局胜负中，我们得到了 Bayes 判别函数。其在训练集上的预测正确率最大为 76.00%，在测试集上的预测正确率最大为75.28%。也就是说我们只通过 14 分钟时的部分时间线数据就能正确预测 75%左右的对局，这个结果还算可以。在线性判别分析模型中，我们只考虑了双方小龙夺取数之差、峡谷先锋夺取数之差、防御塔丢失数以及各位置的对位经济差，而没有考虑双方的英雄选择和阵容。除了这些游戏内的因素外，游戏外的因素也对游戏胜负有很大的影响，如玩家的心态。甚至可以夸张一点地说英雄联盟这款游戏心态比游戏水平更加重要，但这并不是模型所能度量的了。

英雄联盟这个游戏，不只有对线，还有打团。前期对线拿到优势并不等于这把游戏就拿下了，稳扎稳打才能最终拿下胜利；前期劣势也不要摆烂，放平心态，慢慢发育打团，尽力就好。

参考文献

[1] 陈希孺, 倪国熙. 数理统计学教程[M]. 合肥: 中国科学技术大学出版社, 2009, 305-337.

[2] 高惠璇. 应用多元统计分析[M]. 北京: 北京大学出版社, 2005, 175-215.

[3] Fisher, R. A.. The use of multiple measurements in taxonomic problems[J]. Annals of eugenics. 1936, 7(2): 179-188.

[4] Thomas Lumley. The 'iris' data[EB/OL]. https://notstatschat.tumblr.com/post/155194690691/the-iris-data, 2016-12-31.

[5] Altman, Edward I.. Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy[J]. Journal of Finance. 1968, 23(4): 189–209.

[6] 崔自峰, 吉小华. 基于线性判别分析的特征选择[J]. 计算机应用. 2009, 29(10): 5.

[7] James, G.等. 统计学习导论——基于 R 应用[M]. 王星等, 译. 北京: 机械工业出版社，2015, 89-117.

[8] 谢达东, 吴及, 王作英. 线性判别分析在汉语语音识别中的应用[J]. 计算机工程与应用. 2002, 38(023): 1-2,8.

[9] 周大可, 杨新, 彭宁嵩. 改进的线性判别分析算法及其在人脸识别中的应用[J]. 上海交通大学学报. 2005, 39(4): 4.

[10] 韩鸿哲, 王志良, 刘冀伟, 李正熙, 陈锋军. 基于线性判别分析和支持向量机的步态识别[J]. 模式识别与人工智能. 2005, 18(2): 5.

[11] 温俊芹, 王修晖. 基于线性判别分析和自适应 K 近邻法的手势识别[J]. 数据采集与处理. 2017, 32(3): 6.

[12] 霍中花, 陈莹. 采用增量式线性判别分析的行人再识别[J]. 小型微型计算机系统. 2017, 38(3): 6.

[13] Wikipedia, Wikimedia Foundation. League of Legends[EB/OL]. https://en.wikipedia.org/wiki/League_of_Legends, 2022-05-09.

[14] 英雄联盟赛事官方. 《英雄联盟》入选杭州亚运会正式竞赛项目[EB/OL]. https://lol.qq.com/news/detail.shtml?docid=17818141296137619867, 2021-11-05.

[15] LOL planner. 召唤师峡谷地图[Z]. https://map.riftkit.net/, 2022-5-18.

[16] pseudonym117. Welcome to RiotWatcher's documentation![EB/OL]. https://riot-watcher.readthedocs.io/en/latest/, 2022-04-25.

[17] meraki-analytics. Cassiopeia Documentation[EB/OL]. https://cassiopeia.readthedocs.io/en/latest/, 2022-04-15.

附录 1：R 代码

该部分代码已上传至 https://raw.githubusercontent.com/loreliu/bachelor-thesis/main/bachelor-thesis.R.

```
# 本科论文《线性判别分析的原理与应用》用到的代码

# 平台信息及软件包版本
xfun::session_info(c('GGally', 'ggplot2', 'dplyr', 'tidyr', 'forcats',
'viridis',
    'mvnormtest', 'MVTests', 'MASS', 'klaR'))
# R version 4.1.3 (2022-03-10)
# Platform: x86_64-pc-linux-gnu (64-bit)
# Running under: Ubuntu 20.04.4 LTS, RStudio 0

# Locale:
#   LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
LC_TIME=en_US.UTF-8
#   LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
LC_MESSAGES=en_US.UTF-8
#   LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
#   LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
LC_IDENTIFICATION=C

# Package version:
#   base64enc_0.1.3    bit_4.0.4          bit64_4.0.5        bslib_0.3.1
#   cachem_1.0.6       class_7.3.20       classInt_0.4.3     cli_3.3.0
#   clipr_0.8.0        colorspace_2.0.3   combinat_0.0.8
commonmark_1.8.0
#   cpp11_0.4.2        crayon_1.5.1       DEoptimR_1.0.11
digest_0.6.29
#   dplyr_1.0.9        e1071_1.7.9        ellipsis_0.3.2     fansi_1.0.3
#   farver_2.1.0       fastmap_1.1.0      fontawesome_0.2.2
forcats_0.5.1
#   fs_1.5.2           generics_0.1.2     GGally_2.1.2
ggplot2_3.3.6
```

```
#   glue_1.6.2         graphics_4.1.3     grDevices_4.1.3    grid_4.1.3
#   gridExtra_2.3      gtable_0.3.0       haven_2.5.0        highr_0.9
#   hms_1.1.1          htmltools_0.5.2    httpuv_1.6.5
isoband_0.2.5
#   jquerylib_0.1.4    jsonlite_1.8.0     KernSmooth_2.23.20 klaR_1.7.0
#   labeling_0.4.2     labelled_2.9.1     later_1.3.0
lattice_0.20.45
#   lifecycle_1.0.1    magrittr_2.0.3     MASS_7.3.56
Matrix_1.4.1
#   methods_4.1.3      mgcv_1.8.40        mime_0.12
miniUI_0.1.1.1
#   munsell_0.5.0      mvnormtest_0.1.9   MVTests_2.1.1
mvtnorm_1.1.3
#   nlme_3.1.157       pcaPP_2.0.1        pillar_1.7.0
pkgconfig_2.0.3
#   plyr_1.8.7         prettyunits_1.1.1  progress_1.2.2
promises_1.2.0.1
#   proxy_0.4.26       purrr_0.3.4        questionr_0.7.7
R.cache_0.15.0
#   R.methodsS3_1.8.1  R.oo_1.24.0        R.utils_2.11.0     R6_2.5.1
#   rappdirs_0.3.3     RColorBrewer_1.1.3 Rcpp_1.0.8.3       readr_2.1.2
#   rematch2_2.1.2     reshape_0.8.9      rlang_1.0.2
robustbase_0.95.0
#   rprojroot_2.0.3    rrcov_1.7.0        rstudioapi_0.13    sass_0.4.1
#   scales_1.2.0       shiny_1.7.1        sourcetools_0.1.7
splines_4.1.3
#   stats_4.1.3        stats4_4.1.3       stringi_1.7.6
stringr_1.4.0
#   styler_1.7.0       tibble_3.1.7       tidyr_1.2.0
tidyselect_1.1.2
#   tools_4.1.3        tzdb_0.3.0         utf8_1.2.2         utils_4.1.3
#   vctrs_0.4.1        viridis_0.6.2      viridisLite_0.4.0  vroom_1.5.7
#   withr_2.5.0        xfun_0.30          xtable_1.8.4

xfun::pkg_attach2(c('GGally', 'ggplot2', 'dplyr', 'tidyr',
'forcats','viridis'))
```

```
timeline_840 =
read.csv('https://raw.githubusercontent.com/loreliu/bachelor-
thesis/main/timeline_840.csv', header=TRUE,
    encoding='UTF-8')
# timeline_840 = read.csv('timeline_840.csv', header = TRUE, encoding =
'UTF-8')

# 因子化 winningTeam
timeline_840$winningTeam = as.factor(timeline_840$winningTeam)

summary(timeline_840)

# 分组 summary
summary_blue_red = by(timeline_840[, 1:15], timeline_840$winningTeam,
summary)
summary_blue_red

# 分组 std
timeline_840_winningTeam_blue = timeline_840$winningTeam == 100
sd_winblue = lapply(timeline_840[timeline_840_winningTeam_blue, ][,
1:15], sd)
sd_winred = lapply(timeline_840[!timeline_840_winningTeam_blue, ][,
1:15], sd)
sd_winblue
sd_winred


attach(timeline_840)
dragonDiff = t1Dragons - t2Dragons
riftDiff = t1Rift - t2Rift
topTowerTakenDiff = t1TopTowerTaken - t2TopTowerTaken
midTowerTakenDiff = t1MidTowerTaken - t2MidTowerTaken
botTowerTakenDiff = t1BotTowerTaken - t2BotTowerTaken
```

```
summary(cbind(dragonDiff, riftDiff, topTowerTakenDiff,
midTowerTakenDiff, botTowerTakenDiff))
timeline_840_Diff = cbind(timeline_840, cbind(dragonDiff, riftDiff,
topTowerTakenDiff, midTowerTakenDiff,
    botTowerTakenDiff))

# 分组 diff mean
mean_diff_winblue =
lapply(timeline_840_Diff[timeline_840_winningTeam_blue, ][, 28:32],
mean)
mean_diff_winred =
lapply(timeline_840_Diff[!timeline_840_winningTeam_blue, ][, 28:32],
mean)
mean_diff_winblue
mean_diff_winred

# 分组 diff std
sd_diff_winblue =
lapply(timeline_840_Diff[timeline_840_winningTeam_blue, ][, 28:32], sd)
sd_diff_winred =
lapply(timeline_840_Diff[!timeline_840_winningTeam_blue, ][, 28:32], sd)
sd_diff_winblue
sd_diff_winblue


# 各变量的相关系数矩阵
ggcorr(timeline_840[, 1:15], method = c('pairwise', 'pearson'), label =
TRUE, label_alpha = TRUE,
    label_round = 4, hjust = 0.5, name = '各变量的相关系数矩阵')

# 各位置的分组对位经济差散点图矩阵
pairs_diff_gold = ggpairs(timeline_840[, 11:15], ggplot2::aes(color =
winningTeam, alpha = 0.5),
    upper = list(continuous = wrap('cor', size = 2.5)),
    lower = list(continuous = 'smooth'), title='各位置的分组对位经济差散点图
矩阵')
```

```
# 修改调色板 100：blue 200：red
for (i in 1:pairs_diff_gold$nrow) {
    for (j in 1:pairs_diff_gold$ncol) {
        pairs_diff_gold[i, j] = pairs_diff_gold[i, j] +
            scale_fill_manual(values = c('blue', 'red')) +
            scale_color_manual(values = c('blue', 'red'))
    }
}
pairs_diff_gold


# 对位经济领先胜率
win_perc_gold = function(start, end, interval, columns) {
    n = floor((end - start) / interval) # 区间数
    df_prec_gold = data.frame() # 存储计算结果
    intervals = c() # 区间名
    for (i in 1:length(columns)) {
        for (j in 0:n) {
            if (j < n) {
                # 区间名
                if (i == 1) {
                    intervals[length(intervals) + 1] = paste(start +
interval * j, '-', start + interval * (j + 1))
                }
                # 计算胜率
                tl_subset = timeline_840[start + interval * j <=
timeline_840[, columns[i]] & timeline_840[, columns[i]] <
                    start + interval * (j + 1), ]
                prec = sum(tl_subset$winningTeam == 100) /
nrow(tl_subset)
                df_prec_gold[i, (j + 1)] = prec
            } else {
                # 区间名
                if (i == 1) {
                    intervals[length(intervals) + 1] = paste('>', start
+ interval * j)
```

```
                }
                # 计算胜率
                tl_subset = timeline_840[start + interval * j <=
timeline_840[, columns[i]], ]
                prec = sum(tl_subset$winningTeam == 100) /
nrow(tl_subset)
                df_prec_gold[i, (j + 1)] = prec
            }
        }
    }
    rownames(df_prec_gold) = columns
    colnames(df_prec_gold) = intervals
    win_perc_list = list('intervals' = intervals, 'df' = df_prec_gold)
    return(win_perc_list)
}
win_perc_list = win_perc_gold(0, 4500, 500,
colnames(timeline_840)[11:15])
win_perc_list


# 对位经济差折线图
win_perc_df = data.frame(gold=rep(seq(0,4500,500),rep(5,10)),
    win_perc=unname(unlist(win_perc_list$df)),
    role=factor(rep(rownames(win_perc_list$df),5),levels =
rownames(win_perc_list$df)))

ggplot(win_perc_df, aes(x = gold, y = win_perc, group = role, color =
role)) +
    geom_line() +
    scale_color_manual(labels=c('上单', '打野', '中单', 'ADC', '辅助'),
values = c('blue', 'red',
        'magenta', 'green', 'black'))+
    labs(color = '位置') +
    xlab('14 分钟时蓝色方各位置的对位经济领先') +
    ylab('蓝色方各位置的获胜概率') +
    ggtitle('14 分钟时蓝色方各位置对位经济领先的胜率')
```

```
# 蓝色方夺取小龙胜率
df_prec_dragons = data.frame() # 存储计算结果
for(i in 0:2) {
    for(j in 0:2) {
        # 计算胜率
        tl_subset = timeline_840[timeline_840$t1Dragons == i &
timeline_840$t2Dragons == j, ]
        prec = sum(tl_subset$winningTeam == 100)/nrow(tl_subset)
        df_prec_dragons[i+1, j+1] = prec
    }
}
rownames(df_prec_dragons) = c('blue_team_0', 'blue_team_1',
'blue_team_2')
colnames(df_prec_dragons) = c('red_team_0', 'red_team_1', 'red_team_2')
df_prec_dragons


# 蓝色方夺取峡谷先锋胜率
df_prec_rift = data.frame() # 存储计算结果
for(i in 0:1) {
    for(j in 0:1) {
        # 计算胜率
        tl_subset = timeline_840[timeline_840$t1Rift==i &
timeline_840$t2Rift ==j, ]
        prec = sum(tl_subset$winningTeam == 100)/nrow(tl_subset)
        df_prec_rift[i+1, j+1] = prec
    }
}
rownames(df_prec_rift) = c('blue_team_0', 'blue_team_1')
colnames(df_prec_rift) = c('red_team_0', 'red_team_1')
df_prec_rift


# 蓝色方摧毁防御塔胜率
```

```
win_prec_tower = function(role, t1TowerTaken, t2TowerTaken) {
    df_prec_tower = data.frame() # 存储计算结果
    if(role == 'top' | role == 'bot'){
            towers = 4
            }
            else {
               towers = 6
            }
    for(i in 0:towers){
        for(j in 0:towers){
            tl_subset = timeline_840[t2TowerTaken == i & t1TowerTaken ==
j, ]
            prec = sum(tl_subset$winningTeam == 100)/nrow(tl_subset)
            df_prec_tower[i+1, j+1] = prec
        }
        rownames(df_prec_tower)[i+1] = paste('blue_team_', i)
        colnames(df_prec_tower)[i+1] = paste('red_team_', i)
    }
    return(df_prec_tower)
}
win_prec_tower('top', timeline_840$t1TopTowerTaken,
timeline_840$t2TopTowerTaken)
win_prec_tower('mid', timeline_840$t1MidTowerTaken,
timeline_840$t2MidTowerTaken)
win_prec_tower('bot', timeline_840$t1BotTowerTaken,
timeline_840$t2BotTowerTaken)


# 各变量差值的相关系数矩阵
ggcorr(timeline_840_Diff[,c(28:32, 11:15)], method = c('pairwise',
'pearson'), label = TRUE, label_alpha = TRUE,
    label_round = 4, hjust = 0.5, name = '各变量差值的相关系数矩阵')


# 蓝色方获胜时各位置的对位经济差直方图
```

```
tl_gold_diff_blue =
timeline_840_Diff[timeline_840_Diff$winningTeam==100,c(11:15)] %>%
    gather(key = 'text', value = 'value') %>%
    mutate(text = gsub('\\.', ' ', text))

tl_gold_diff_blue_plot = tl_gold_diff_blue %>%
    mutate(text = fct_reorder(text, value)) %>%
    ggplot(aes(x = value, color = text, fill = text)) +
        geom_histogram(alpha = 0.5, binwidth = 5) +
        scale_fill_viridis(discrete = TRUE) +
        scale_color_viridis(discrete = TRUE) +
        theme(
        legend.position = 'none',
        panel.spacing = unit(0.1, 'lines'),
        strip.text.x = element_text(size = 8)
        ) +
        xlab('对位经济差') +
        ylab('蓝色方获胜时各位置的对位经济差直方图') +
        facet_wrap(~text)
tl_gold_diff_blue_plot


# 紫色方获胜时各位置的对位经济差直方图
tl_gold_diff_red = timeline_840_Diff[timeline_840_Diff$winningTeam ==
200, c(11:15)] %>%
    gather(key = 'text', value = 'value') %>%
    mutate(text = gsub('\\.', ' ', text))

tl_gold_diff_red_plot = tl_gold_diff_blue %>%
    mutate(text = fct_reorder(text, value)) %>%
    ggplot(aes(x = value, color = text, fill = text)) +
        geom_histogram(alpha = 0.5, binwidth = 5) +
        scale_fill_viridis(discrete = TRUE) +
        scale_color_viridis(discrete = TRUE) +
        theme(
        legend.position='none',
```

```
        panel.spacing = unit(0.1, 'lines'),
        strip.text.x = element_text(size = 8)
        ) +
        xlab('对位经济差') +
        ylab('紫色方获胜时各位置的对位经济差直方图') +
        facet_wrap(~text)
tl_gold_diff_red_plot


# Shapiro-Wilk 检验
mvnormtest::mshapiro.test(t(as.matrix(timeline_840_Diff[timeline_840_Dif
f$winningTeam == 100, c(11:15)])))
mvnormtest::mshapiro.test(t(as.matrix(timeline_840_Diff[timeline_840_Dif
f$winningTeam == 200, c(11:15)])))


# BoxM 检验
MVTests::BoxM(timeline_840_Diff[, c(28, 29, 5:15)],
timeline_840_Diff$winningTeam)
MVTests::BoxM(timeline_840_Diff[, c(28, 29, 11:15)],
timeline_840_Diff$winningTeam)
MVTests::BoxM(timeline_840_Diff[, c(5:10)],
timeline_840_Diff$winningTeam)


# 为了使结果可复现 从 random.org 获取一个随机数作为种子
set.seed(4225)

# 将数据集分为训练集和测试集，比例为 8:2
index_train = sample(c(TRUE, FALSE), nrow(timeline_840_Diff), replace =
TRUE, prob = c(0.8, 0.2))
training = timeline_840_Diff[index_train, ]
testing = timeline_840_Diff[!index_train, ]

# 建立 Bayes 判别模型
```

```
lda_fit = MASS::lda(winningTeam ~ dragonDiff + riftDiff +
t1TopTowerTaken + t2TopTowerTaken + t1MidTowerTaken +
    t2MidTowerTaken + t1BotTowerTaken + t2BotTowerTaken + topGoldDiff +
jugGoldDiff + midGoldDiff +
    adcGoldDiff + supGoldDiff, prior = c(0.5, 0.5), data = training)

str(lda_fit)
lda_fit

# 训练集混淆矩阵
table(training$winningTeam, predict(lda_fit)$class)
# 训练集预测正确率
sum(training$winningTeam == predict(lda_fit)$class)/nrow(training) #
0.7578272

# 测试集混淆矩阵
table(testing$winningTeam, predict(lda_fit, newdata = testing)$class)
# 测试集预测正确率
sum(testing$winningTeam == predict(lda_fit, newdata =
testing)$class)/nrow(testing)


# 找出训练集预测正确率最大的临界值
critical = function(dataframe){
    prec_df = data.frame(critical = NA, prec = NA)
    j = 0
    for(i in seq(0.01, 0.99, 0.01)) {
        j=j+1
        if(identical(dataframe, training)){
            confus = table(dataframe$winningTeam,
predict(lda_fit)$posterior[ , '100'] >= i)
        }
        else{
            confus = table(dataframe$winningTeam, predict(lda_fit,
newdata = dataframe)$posterior[ , '100'] >= i)
        }
```

```
        prec = (confus[, 2][1] + confus[, 1][2])/sum(confus)
        prec_df[j, ] = c(i, prec)
    }
    return(prec_df[which.max(prec_df$prec), ])
}
critical(training)
critical(testing)


# Bayes（Fisher）判别函数的直方图
plot(lda_fit ,dimen = 1, type = 'b')

# Bayes 判别函数在测试集对位经济差上的分区图
klaR::partimat(winningTeam ~ topGoldDiff + jugGoldDiff + midGoldDiff +
adcGoldDiff + supGoldDiff,
    data = testing, method = 'lda')


# 测试集正判样品
head(testing[testing$winningTeam == predict(lda_fit, newdata =
testing)$class, ])
correct_testing_summary = summary(testing[testing$winningTeam ==
predict(lda_fit, newdata = testing)$class, ])
# 测试集预测正确时游戏时长的统计特征
# correct_testing_summary[,16]
summary(testing[testing$winningTeam == predict(lda_fit, newdata =
testing)$class, 'gameLength'])


# 测试集误判样品
head(testing[!(testing$winningTeam == predict(lda_fit, newdata =
testing)$class), ])
error_testing_summary = summary(testing[!(testing$winningTeam ==
predict(lda_fit, newdata = testing)$class), ])
# 测试集预测错误时游戏时长的统计特征
# error_testing_summary[,16]
```

```
summary(testing[!(testing$winningTeam == predict(lda_fit, newdata =
testing)$class), 'gameLength'])


# 测试集预测正确和预测错误时游戏时长的小提琴图
correct_bayes_gl = testing[testing$winningTeam == predict(lda_fit,
newdata = testing)$class, 'gameLength']
error_bayes_gl = testing[!(testing$winningTeam == predict(lda_fit,
newdata = testing)$class), 'gameLength']
testing_bayes_gameLength = data.frame(
    bayes_gameLength = c(correct_bayes_gl, error_bayes_gl),
    results = c(rep('预测正确', length(correct_bayes_gl)), rep('预测错误',
length(error_bayes_gl)))
)

ggplot(testing_bayes_gameLength, aes(x = results, y = bayes_gameLength,
fill = results)) +
    geom_violin() +
    scale_fill_manual(labels = c('预测正确', '预测错误'), values =
c('blue', 'red'))+
    labs(fill = '结果') +
    xlab('Bayes 判别预测结果') +
    ylab('游戏时长/秒') +
    ggtitle('测试集预测正确和预测错误时游戏时长的小提琴图')
```

附录 2：开题报告

开题报告

致谢

感谢 Riot Games API 的 Python 封装库 Riot-watcher（https://github.com/pseudonym117/Riot-Watcher）和 Cassiopeia（https://github.com/meraki-analytics/cassiopeia）的贡献者们。感谢帮我检查论文的王倩老师，拖了这么久，实在抱歉。

和四年前刚刚经历高考时的自己相比，大学这四年朝不同方向努了很久，收获了很多东西，但也在不经意间丢掉了一些东西，如耐心和专注力。或许是和高中时期的时间分配有关，虽然大学有更多的时间可以投入到自己感兴趣的地方，但是不像在高中时期可以专注于一件事情很久。收获有很多，最大的收获就是遇到了统计，由此认识了一群统计人，他们的经历、人生观时时刻刻在激励着我，让我知道怎样去做一个更好的人。感谢你们让我成为了一个喜欢分享，热爱奉献的人；也感谢概率和统计，帮助我更好地理解人生、理解这个世界，给了我很多解决问题的答案，从某种意义上来讲，我们每个人的整个人生都包含在里面。

最后借用陈大岳老师的一句名言“人生不是平稳过程，但遍历定理仍然成立”来结束这篇致谢。